POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Confection de tournées de livraison dans un réseau urbain à l'aide de métaheuristiques et de méthodes de forage de données massives

MAHA GMIRA
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de Philosophiæ Doctor
Génie industriel

Octobre 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Confection de tournées de livraison dans un réseau urbain à l'aide de métaheuristiques et de méthodes de forage de données massives

présentée par **Maha GMIRA**
en vue de l'obtention du diplôme de Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

Louis-Martin ROUSSEAU, président
Michel GENDREAU, membre et directeur de recherche
Andrea LODI, membre et codirecteur de recherche
Jean-Yves POTVIN, membre et codirecteur de recherche
Martin TRÉPANIER, membre
Dominique FEILLET, membre externe

REMERCIEMENTS

Je remercie mes directeurs, Michel Gendreau, Jean-Yves Potvin et Andrea Lodi, qui m'ont beaucoup apporté d'un point de vue scientifique, méthodologique et humain, aussi, pour leur expertise, leur confiance, leur engagement et leur disponibilité. Particulièrement Michel Gendreau qui m'a accordé sa confiance, m'a enseigné l'importance de la patience et de la curiosité et surtout m'a proposé un sujet des plus intéressants. Michel a toujours été d'un très grand apport tout au long de nos rencontres. Également, Jean-Yves Potvin, pour sa grande minutie, sa méthodologie et nos échanges qui m'ont permis de m'améliorer comme chercheuse, et finalement Andrea Lodi qui m'a appris la persévérance dans la recherche scientifique et qui m'a accordé sa confiance.

Je remercie également l'équipe administrative et technique : Khalid et Koladé, pour leurs écoute et aide, de belles amitiés en perspective, Lucie et Lucie-Nathalie pour nos agréables discussions et leurs bonnes humeurs, à Serge et Pierre pour leurs aides précieuses pour surmonter les difficultés informatiques.

Enfin, je remercie ma famille : d'abord mes parents, sans qui je n'en serai probablement pas là aujourd'hui. Ils m'ont écouté, épaulé et conseillé tout au long de ma vie, et tout particulièrement, durant mes études. Pour leur amour, leur tendresse et leur bienveillance, je leur dédie ce manuscrit de thèse et le travail de ces dernières années.

Mes soeurs, Chaimae et Imane, pour leurs amours, leurs attentions et nos grands moments de complicité. Pour tout cela et bien d'autres, je vous aime.

Une dédicace spéciale pour mon mari, Badr, qui a été ma force dans les moments difficiles. Il a su faire preuve de beaucoup de compréhension lors de longues périodes de travail acharné, a su s'intéresser à mes travaux de recherche et m'écouter parler de mon sujet de thèse de longues heures, malgré la différence de nos domaines d'expertise, a su trouver les mots justes pour me redonner la motivation nécessaire.

Une pensée pour mes nombreux amis et collègues de la chaire d'excellence en recherche du Canada sur la science des données pour la prise de décision en temps réel et de Polytechnique, pour les bons moments passés ensemble et nos belles discussions.

Enfin, je tiens à remercier tous les membres du jury d'avoir accepté d'évaluer ma thèse.

RÉSUMÉ

La confection de tournées de livraison dans un réseau urbain est un problème qui peut faire appel à la recherche opérationnelle, au forage de données, à l'intelligence artificielle et à la géomatique, entre autres disciplines. Il s'agit de trouver une affectation des clients aux véhicules de livraison et un ordre de visite des clients pour chacun des véhicules afin d'optimiser une certaine fonction objectif. La considération d'une version dynamique du problème, où le processus d'optimisation doit aussi tenir compte de nouvelles données qui affluent en temps réel, rend la résolution encore plus complexe. De tels problèmes se retrouvent, par exemple, dans les entreprises qui offrent des services de livraison à domicile. Dans un contexte d'intégration, de globalisation et de compétitivité, ces entreprises se doivent de prendre les décisions menant aux meilleures tournées de livraison possibles. La recherche d'une solution optimale est souvent impossible pour des instances de taille réaliste. Afin d'obtenir de bonnes solutions dans des temps de calcul raisonnables, des approches heuristiques et métaheuristiques ont été proposées dans la littérature. Cependant, la majorité des travaux se basent sur des instances synthétiques qui, entre autres, négligent la topologie des réseaux routiers rencontrés dans la pratique.

Cette thèse vise, dans un premier temps, à contribuer à l'intégration des méthodes de forage de données et d'apprentissage automatique dans la prédiction des vitesses de parcours dans un réseau routier réel. Grâce à la collaboration d'un partenaire industriel, nous disposons d'une base de données de points GPS recueillis dans la région métropolitaine de Montréal à partir de dispositifs embarqués dans des véhicules de livraison, couvrant une période de plus de deux ans. Nous proposons une méthode de prédiction des vitesses sur les arcs de ce réseau en faisant appel au forage de données massives et à l'apprentissage automatique : techniques de réduction de dimensionnalité, méthodes d'imputation et d'apprentissages supervisé et non supervisé. L'ensemble de cette méthodologie a fait l'objet d'un article, actuellement en révision, soumis à la revue scientifique *EURO Journal on Transportation and Logistics*.

Cette thèse vise aussi à contribuer à l'avancement des métaheuristiques pour la résolution de problèmes de tournées de livraisons à domicile définies sur un réseau routier, avec des vitesses et temps de parcours qui dépendent du moment de la journée, des fenêtres de temps pour le service aux clients et une capacité maximale pour les véhicules. Plus précisément, nous proposons une recherche tabou qui remet en question l'ordre de visite des clients ainsi que le chemin utilisé dans le réseau routier pour se rendre d'un client à un autre, en fonction du moment de la journée. Un développement majeur de ce travail est la conception de

techniques d'évaluation en temps constant de la réalisabilité ainsi que du coût approximatif des solutions dans le voisinage de la solution courante. Ces techniques permettent de réduire l'effort computationnel et de résoudre des instances avec 200 nœuds et 580 arcs dans des temps de calcul très raisonnables. L'approche de résolution incluant la description des techniques d'évaluation en temps constant de la réalisibilité des solutions dans le voisinage et de leurs coûts approximatifs a fait l'objet d'un article soumis à la revue scientifique *Transportation Science*.

Enfin, la thèse s'attaque à une variante dynamique du problème précédent, dans un contexte où les vitesses sur les arcs du réseau routier varient de façon dynamique due à des incidents imprévus. Un simulateur a été développé pour générer les perturbations dynamiques aux vitesses sur les arcs, tout en respectant les relations spatio-temporelles entre ces derniers. Une procédure permet ensuite de réagir aux perturbations en modifiant la solution courante de façon plus ou moins importante selon l'importance des perturbations. La démarche développée pour résoudre ce problème dynamique de confection de tournées qui inclut l'ajustement à des plans de livraison déjà optimisés dans un contexte statique est présentée dans un article ayant été soumis à la revue scientifique *European Journal of Operational Research*.

ABSTRACT

Fleet management for home deliveries in an urban context is at the crossroads between several disciplines such as operations research, data mining, artificial intelligence, geomatic, etc. The objective is to find 1) assignment of customers to vehicles and 2) sequence of customers visited by each vehicle to optimize a certain objective function.

Since accurate travel time predictions are of foremost importance in urban environments, it is important to address dynamic variants of vehicle routing problems, because they better model the problems faced by transportation companies like home delivery services. In an era where companies have to integrate their supply chain, face globalization and competitiveness, it is important for them to make decisions leading to the best possible delivery routes.

Therefore, in this thesis, we first consider the prediction of travel speeds using as input GPS traces of commercial vehicles collected over a significant period of time. We propose a forecasting framework based on machine learning and data mining techniques: dimensionality reduction techniques, imputation methods, unsupervised and supervised learning.

Secondly, we propose a solution approach to solve a time-dependent vehicle routing problem with time windows in which travel speeds are defined on the road network itself. The solution approach involves a tabu search heuristic that considers different shortest paths between pairs of customers at different times of the day. A major contribution of this work is the development of techniques to evaluate the feasibility as well as the approximate cost of a solution in constant time, thus allowing the overall solution approach to handle instances with up to 200 nodes and 580 arcs in very reasonable computing times.

Finally, we consider a dynamic vehicle routing problem motivated from home delivery applications that can adapt to changes in speeds by modifying the paths used in the road network to go from one customer to the next and by modifying the sequences of customers in the planned routes. Here, uncertainty comes from a single source, namely, the occurrence of new traffic information that affects speeds. To mimic real-time perturbations in the road network, we developed a simulator that generates traffic events and updates the speeds accordingly.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AP	Affinity Propagation
ARIMA	AutoRegressive Integrated Moving Average
ARMA	AutoRegressive Moving Average
CPU	Central Processing Unit
DVRP	Dynamic Vehicle Routing Problem
EKF	Extended Kalman Filter
EM	Expectation Maximization
FIFO	First-In First-Out
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
IA	Intelligence Artificielle
k-NN	k-Nearest Neighbor
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MI	Multiple Imputation
MICE	Multivariate Imputation via Chained Equation
MILP	Mixed Integer Linear Program
MLP	Multi-Layer Perceptron
NC	Non-Correlated
NPR	Non-Parametric Regression
NTW	Narrow Time Window
RF	Random Forest
RME	Relative Mean Error
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SARIMA	Seasonal AutoRegressive Integrated Moving Average
SC	Strongly Correlated
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
SVR	Support Vector Regression
TDSPP	Time-Dependent Shortest Path Problem
TDVRP	Time-Dependent Vehicle Routing Problem

$TDVRPTW_{RN}$	Time-Dependent Vehicle Routing Problem with Time Windows on a Road Network
TS	Tabu Search
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
WC	Weakly Correlated
WTW	Wide Time Window

CHAPITRE 1 INTRODUCTION

1.1 Motivation

Le transport par camion revêt une importance particulière, notamment dans un grand pays comme le Canada, qui compte plus d'un million de kilomètres de routes, dont environ 38000 kilomètres formant le réseau routier national. Le transport routier est le mode le plus important pour le transport des personnes et des marchandises, le transport local et interurbain, les activités de transport intraprovinciales, ainsi que les échanges commerciaux entre le Canada et les États-Unis (au titre de la valeur transportée). L'importance de l'industrie canadienne du transport des marchandises pour le compte d'autrui se traduit par quelques chiffres :

- en 2011, les transporteurs canadiens ont déplacé l'équivalent de 225 milliards de tonnes-kilomètres de marchandises, en hausse de 8,1 % par rapport à 2010. Environ 139 milliards de tonnes-kilomètres (61,5 %) ont été transportées dans le secteur intérieur et 87 milliards de tonnes-kilomètres dans le secteur international.
- le transport de marchandises par route comptait pour 63,7 Mt d'eCO_2 (mégatonnes d'équivalents de dioxyde de carbone), représentant 37 % des émissions des transports au Canada en 2008.

De plus, selon une étude récente parue en 2015 [3] portant sur la congestion routière dans les grandes zones urbaines au Canada, Montréal occupe le deuxième rang des villes les plus congestionnées, derrière Toronto. La somme des impacts calculée pour la ville de Montréal s'établirait à 1,7 milliard. Dans cette étude, les coûts annuels directs de la congestion routière sont estimés, sur le plan économique, en fonction des coûts d'opportunité inhérents aux heures perdues et, sur le plan environnemental, en fonction de la consommation excédentaire de carburant (émissions totales de CO_2).

Sachant que les consommateurs assument de 60 à 80% des coûts additionnels qu'entraîne la congestion pour la livraison des marchandises, tandis que le reste est absorbé par les entreprises elles-mêmes [4] et que les émissions de gaz à effet de serre résultant d'un surplus de carburant consommé dans les embouteillages représentent 0,2% du total des émissions canadiennes de gaz à effet de serre, selon Environnement Canada, il est clair que la congestion nuit au déplacement des personnes et des marchandises dans les zones urbaines; or, il est vital pour la compétitivité et la prospérité du Canada que les marchandises soient déplacées de façon efficace. D'autant plus que les coûts exorbitants liés à la congestion sont assumés autant

par les automobilistes que par les entreprises qui souffrent des délais accrus de livraison, du retard de leurs employés et du stress qui en découle.

Heureusement, cette situation peut être améliorée notamment grâce à l'exploitation des grands volumes de données dans le domaine du transport, par le biais de l'Intelligence Artificielle (IA) et de techniques avancées d'optimisation. Ces deux domaines offrent ainsi des pistes de solutions fort prometteuses.

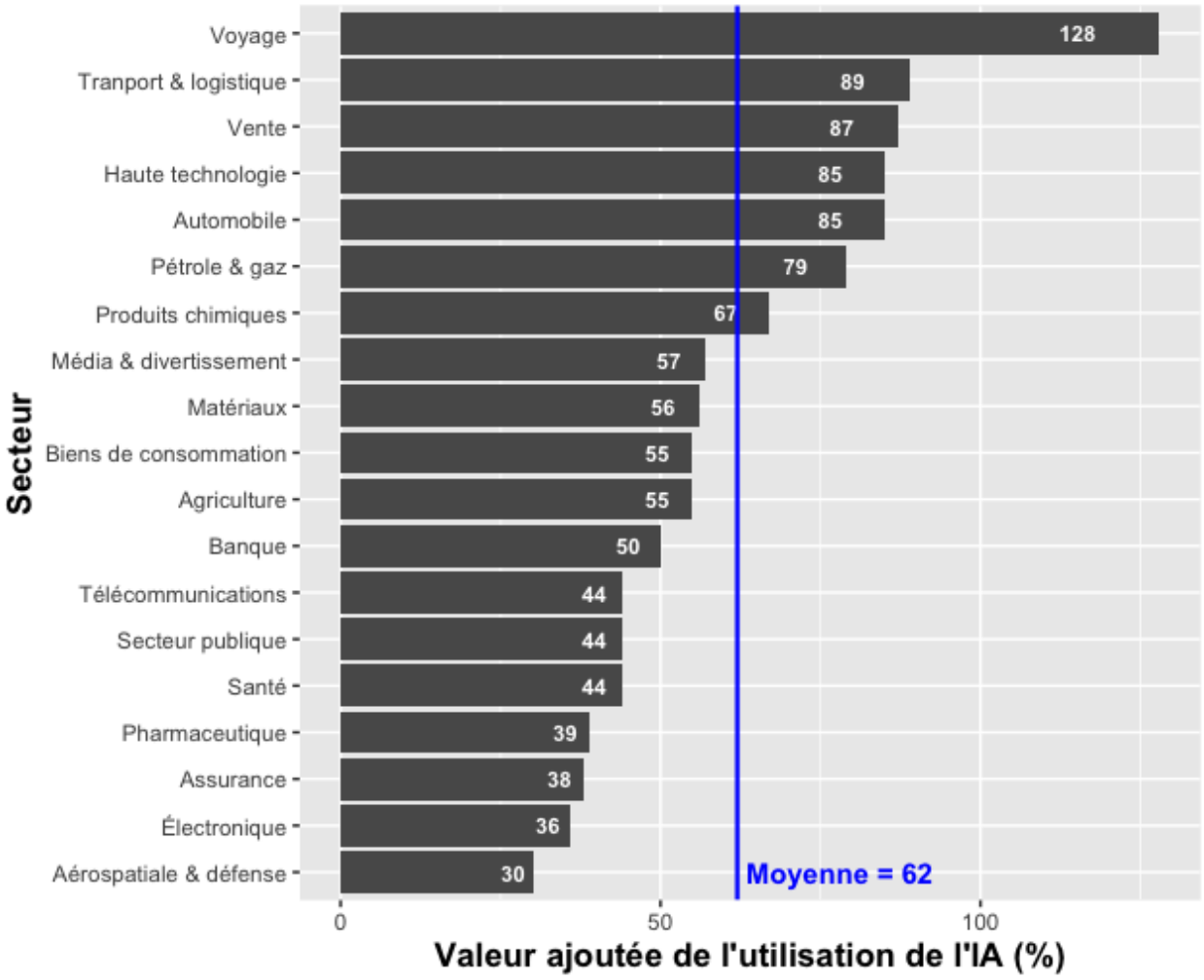


Figure 1.1 Contribution de l'IA dans l'amélioration de différents secteurs – adaptée de [1]

Par exemple, tel que relevé dans un rapport récent publié par McKinsey Global Institute [1], l'IA peut créer de la valeur ajoutée dans de nombreux secteurs, tel qu'illustré à la figure 1.1. Cette étude se base sur 400 projets menés dans différentes entreprises et organisations pour comparer l'utilisation de l'IA, caractérisée par les techniques d'*apprentissage profond*

utilisant des réseaux de neurones artificiels avec d'autres méthodes statistiques.

Les auteurs de cette étude notent également que l'apprentissage profond en IA est en mesure de produire une meilleure valeur ajoutée que des techniques plus traditionnelles. La figure 1.1 montre que le voyage et le transport sont les secteurs où les bénéfices sont les plus significatifs. À titre d'exemple, les techniques d'apprentissage automatique peuvent être utilisées pour trouver un itinéraire plus rapide et plus pratique pour les usagers de la route, ce qui inclut les véhicules de livraison. De plus, le secteur du transport a bénéficié du développement d'une multitude d'outils de collecte de données permettant d'accumuler de grands volumes de données (Big Data), rendant ainsi leur exploitation par les techniques de l'IA très attrayante, autant pour les entreprises que pour les organisations gouvernementales. Étant donné que le terme IA fait référence à un concept en constante évolution, et ce depuis plusieurs décennies, il est donc plus approprié de parler plutot de techniques d'*apprentissage automatique*.

La recherche opérationnelle, de son côté, est une discipline dont les origines remontent à la Seconde Guerre mondiale et qui a permis d'optimiser des problèmes de logistique militaire, entre autres. Après la guerre, la recherche opérationnelle a toutefois vu son champ d'application s'étendre rapidement au monde civil. Les succès de la recherche opérationnelle ne se sont pas démentis au fil des années et elle est devenue un outil largement utilisé pour améliorer l'efficacité des entreprises et des organismes gouvernementaux. La recherche opérationnelle repose sur :

- les mathématiques appliquées;
- les sciences de la gestion;
- l'informatique;
- le génie.

et permet une évaluation quantitative des politiques et des stratégies dans le cadre des opérations d'une organisation. En particulier, la recherche opérationnelle s'intéresse à l'optimisation de problèmes dans le secteur des transports par la modélisation du comportement des usagers, la conception de réseaux d'infrastructure ou de services, la planification de tournées de véhicules, etc. L'utilisation des méthodes de la recherche opérationnelle permet de trouver des solutions optimales à des problèmes complexes en présence de plusieurs contraintes, mais aussi des solutions de très bonne qualité pour des problèmes où l'optimalité n'est pas envisageable, à cause de la taille ou de la complexité trop grande des problèmes.

Cette thèse exploite donc 1) la richesse des données disponibles en transport en faisant appel aux techniques d'analyse de l'IA et 2) la puissance des algorithmes d'optimisation de la recherche opérationnelle. Plus précisément :

- Les différentes méthodes d'apprentissage automatique exploitées dans le cadre de cette thèse ont servi à prédire les vitesses sur les arcs du réseau de Montréal (figure 1.2) à différents moments de la journée. Ainsi, des méthodes d'apprentissages supervisé et non supervisé sont utilisées pour prédire l'évolution des vitesses de parcours sur les arcs en tenant compte de leurs relations spatio-temporelles. Des tests de prédiction ont été menés et des mesures de performances ont été calculées pour démontrer l'intérêt de cette approche. Contrairement à de nombreux travaux qui présentent uniquement le développement de méthodes de prédiction de vitesses de parcours, le chapitre 4 présente une méthodologie complète, capable de traiter une base de données GPS, d'y appliquer différentes méthodes de forage de données et de techniques d'apprentissage automatique.

- La recherche opérationnelle nous a permis de résoudre un problème de confection de tournées de véhicules pour un service de livraison à domicile défini sur un réseau routier avec des vitesses sur les arcs du réseau qui varient en fonction du temps, des clients qui doivent être servis à l'intérieur de fenêtres de temps et des véhicules identiques de capacité finie. Il s'agit alors de déterminer une séquence de clients ou de segments routiers (au niveau du réseau) pour chacun des véhicules de façon à obtenir des itinéraires, commençant et se terminant à un dépôt central, de coût minimal. Les itinéraires doivent tenir compte des vitesses et temps de parcours dépendants du temps ainsi que des contraintes de fenêtres de temps et de capacité des véhicules. Ce type de problème est particulièrement difficile à résoudre et nécessite des temps de calcul conséquents même pour des instances de taille relativement modeste.

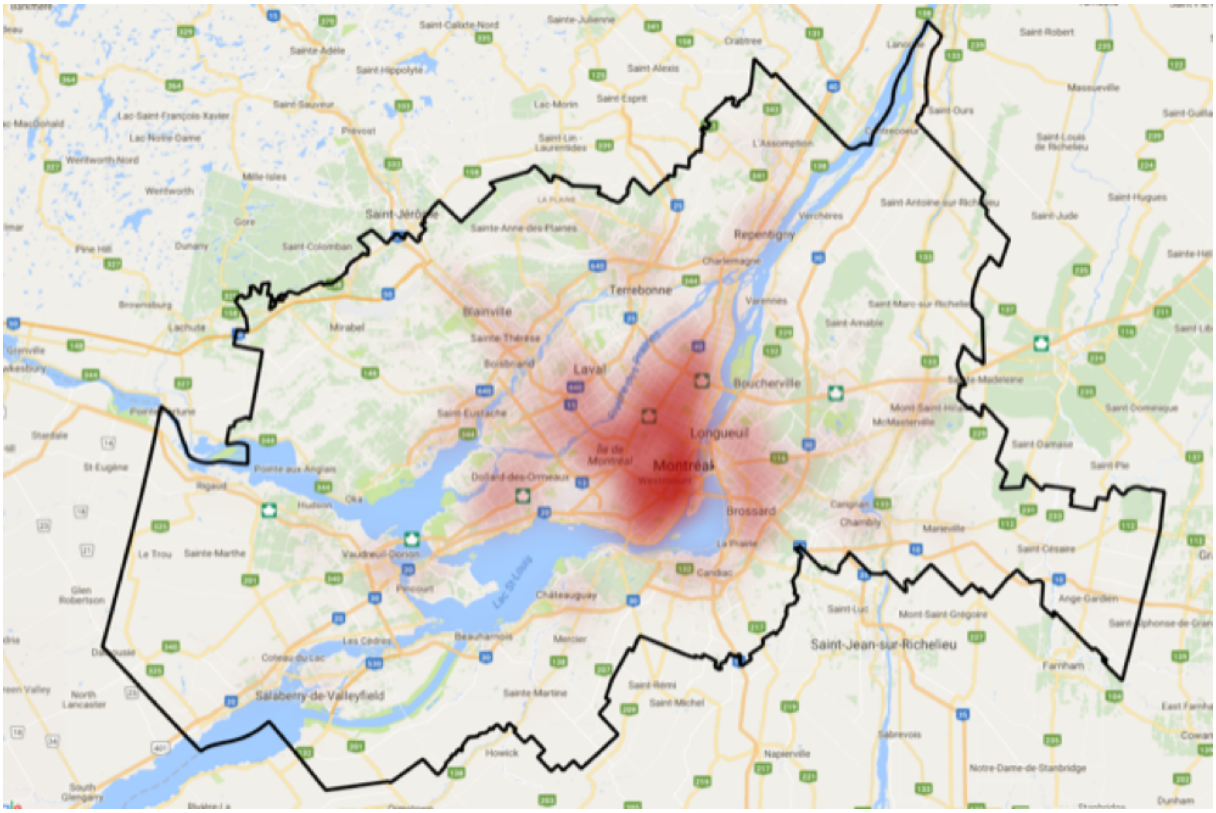


Figure 1.2 La grande région de l'île de Montréal

1.2 Objectifs de recherche

L'objectif de cette thèse est de résoudre un problème réaliste de livraison à domicile qui tient également compte des caractéristiques d'un véritable réseau urbain. Nous abordons d'abord l'estimation et la prédiction des vitesses sur les arcs d'un réseau urbain en fonction du moment de la journée, qui représenteront ensuite des données fournies en entrée à nos algorithmes d'optimisation. Ensuite, nous développons des approches de résolution pour le problème de livraison, tel que défini plus haut. Nous voulons que les méthodes de résolution fournissent des solutions de très bonne qualité dans des délais de calcul raisonnables. L'objectif général de cette thèse se décline ainsi en trois sous-objectifs.

- Le premier sous-objectif s'intéresse à la prédiction, à partir de données GPS, de vitesses et temps de parcours qui dépendent du moment de la journée. Nous exploitons également d'autres variables exogènes, telles que la journée de la semaine, la saison, etc. Nous cherchons donc à développer une structure de traitement de données, basée sur des techniques de forage de données, de réduction de la dimensionnalité et d'apprentissage

automatique pour prédire des vitesses sur les arcs du réseau à partir d'informations extraites de données historiques ainsi que d'autres variables explicatives.

- Le second sous-objectif consiste à développer une approche de nature métaheuristique, plus précisément une recherche tabou, pour résoudre notre problème de livraison à domicile dans un contexte statique où toutes les données pertinentes sont connues à l'avance et ne changent pas. Afin d'être en mesure d'aborder des instances de grande taille, nous ne recherchons pas ici l'optimalité. De plus, pour accélérer les temps de calcul, des techniques ont été développées afin d'évaluer en temps constant la réalisabilité et le coût approximatif des solutions dans le voisinage de la solution courante. Sur la base de cette évaluation approximative, les meilleures solutions dans le voisinage sont sélectionnées et sont ensuite évaluées de manière exhaustive, de façon à identifier la solution dans le voisinage qui deviendra la nouvelle solution courante.

- Le troisième sous-objectif consiste à résoudre une variante dynamique du problème de livraison à domicile où les vitesses varient dynamiquement au cours de la journée, suite à des incidents sur le réseau routier. L'approche de résolution permettant de réagir à de tels événements se doit ainsi de produire des solutions bien adaptées à la nouvelle réalité et qui requièrent des temps de calcul suffisamment courts pour répondre aux exigences d'un environnement qui évolue en temps réel.

1.3 Plan de la thèse

Le chapitre 2 présente une revue de littérature des différents axes de recherche impliqués dans nos contributions. Le chapitre 3 décrit la démarche générale de la thèse et son organisation en trois articles. Ces articles sont ensuite présentés dans les chapitres 4, 5 et 6. Le chapitre 7 propose une discussion générale des différentes approches développées en regard des aspects méthodologiques et des résultats parus dans la revue de littérature. Finalement, la conclusion est présentée dans le chapitre 8, suivie d'une brève analyse de futures avenues de recherche.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans le présent chapitre, le lien entre les trois contributions est mis en évidence à l'aide d'une chronologie de travaux de recherche pertinents. La revue de littérature présente un survol des principales méthodes de résolution proposées en IA ainsi qu'en optimisation statique et dynamique. Comme illustré dans la figure 2.1, cette thèse met en relation ces différentes disciplines.

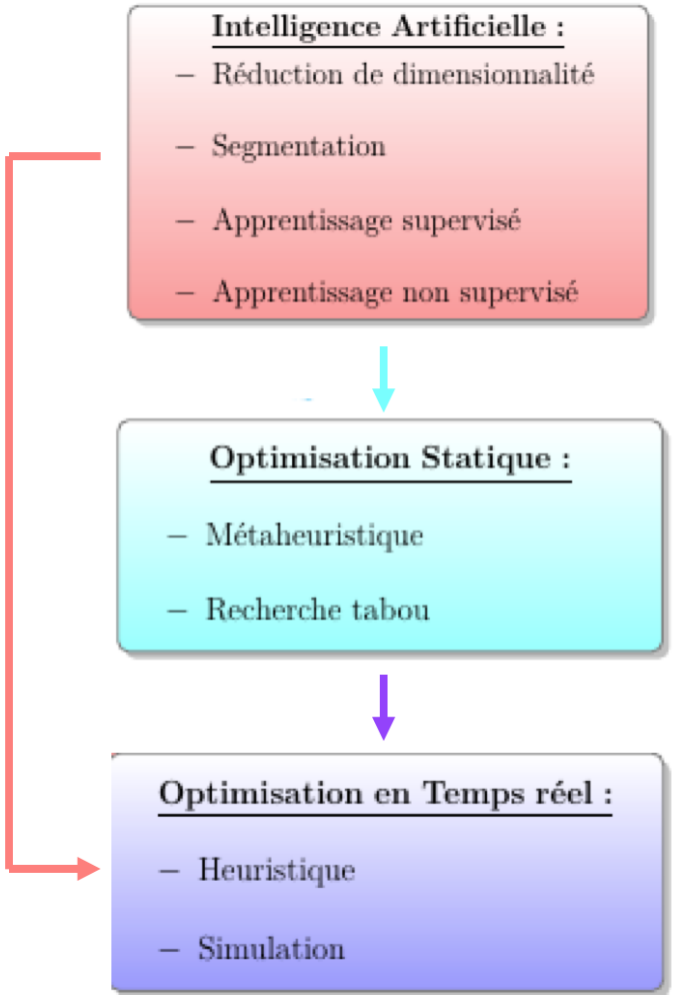


Figure 2.1 Relations entre les axes méthodologiques utilisés dans cette thèse

Chacun des axes de recherche représentés dans la figure 2.1 contient les principales méthodes utilisées dans le cadre de cette thèse. Leurs interactions sont également illustrées à l'aide de

flèches. À titre d'exemple, le pouvoir prédictif de certaines méthodes d'IA alimente en données les méthodes d'optimisation de la recherche opérationnelle (dans les contextes statiques et dynamiques). De même, les méthodes d'optimisation pour la résolution de problèmes statiques offrent souvent un support pour la résolution des variantes dynamiques. En effet, un problème dynamique peut être abordé comme une suite de problèmes statiques définis sur un horizon de temps roulant, où un nouveau problème statique est défini et résolu à chaque fois qu'une mise à jour survient. Ainsi, les trois axes de recherche interagissent de manière cohérente et efficace permettant de bénéficier de chaque méthode utilisée. Il est à noter que, dans cette thèse, chacun de ces axes a fait l'objet d'un article.

2.1 Intelligence Artificielle

L'application de l'IA au secteur du transport permet entre autres de répondre à une demande croissante en modélisation, planification et prise de décision, mais également une volonté de réduire les émissions de CO_2, d'assurer la sécurité routière (véhicules connectés et autonomes), etc. Pour mieux comprendre l'évolution des méthodes d'IA, un bref tour d'horizon est nécessaire. En 1956, le terme IA est utilisé pour la première fois par John McCarthy lors d'une conférence tenue au Dartmouth College, NH. Les ambitions des systèmes d'IA développés à cette époque n'ont toutefois pas été à la hauteur des attentes. Par la suite, soit entre 1960 et 1970, des chercheurs en IA ont exploré des systèmes basés sur les connaissances aussi appelés systèmes experts (traduction des termes anglais knowledge-based systems et expert systems) ainsi que les réseaux de neurones artificiels. Les réseaux de neurones ont connu un intérêt grandissant entre les années 1980 à 1990, grâce à un algorithme d'apprentissage basé sur la rétropropagation du gradient de l'erreur entre les réponses produites par le réseau de neurones et les réponses attendues. La fin des années 90 a été le témoin d'une transformation des méthodes de l'IA, s'orientant désormais davantage vers une base mathématique plus rigoureuse, des applications réelles et le développement de systèmes intelligents hybrides utilisant plus d'une méthode d'IA.

Le secteur du transport possède plusieurs caractéristiques qui permettent d'obtenir des résultats très prometteurs lorsque des méthodes d'apprentissage automatique sont appliquées. Mentionnons, en particulier, la qualité et le volume des données qualitatives et quantitatives, des comportements difficiles à modéliser avec des approches traditionnelles, soit en raison de nombreuses interactions avec d'autres systèmes ou bien à cause de l'incertitude provenant de la composante humaine d'un système.

Malgré les nombreuses applications de l'apprentissage automatique en transport, nous allons nous concentrer dans ce qui suit sur les méthodes de prédiction afin de comprendre comment

l'apprentissage automatique peut être exploité pour appréhender l'évolution des données sur le trafic dans un réseau routier, offrir un meilleur choix d'itinéraires et aider les gestionnaires à mieux allouer leurs ressources en fonction du moment de la journée. Nous nous intéressons ici à deux familles de méthodes de prédiction soit les **méthodes paramétriques** et les **méthodes non paramétriques**.

Méthodes paramétriques.

Les auteurs de [5] utilisent le filtre de Kalman pour prédire les temps de parcours à partir de données synthétiques. Ce modèle a ensuite été réévalué par [6] en utilisant cette fois des données historiques et en temps réel issues de dispositifs d'identification automatique installés dans les autoroutes. Ils concluent que la précision de la prédiction dépend à la fois de la couverture du réseau routier par les véhicules, mais également du niveau de congestion.

Dans [7], les auteurs ont utilisé des données issues à la fois de capteurs embarqués dans les véhicules et de capteurs dans la chaussée pour développer un modèle linéaire pour prévoir les temps de parcours sur les autoroutes. Pour une période de prédiction variant de 0 à 30 minutes, les résultats ont démontré que l'erreur de prédiction varie entre 5% et 13%. Cependant, l'effet de la congestion n'est pas pris en compte, ce qui représente une grande limitation dans un contexte où les temps de parcours varient grandement en fonction du temps.

Dans [8], ils combinent un modèle de régression linéaire à un modèle bayésien afin d'améliorer la précision de la prédiction et sa fiabilité. Cette combinaison de modèles a fourni des résultats précis.

Dans [9], les auteurs ont examiné les profils journaliers des temps de parcours et ils ont appliqué une régression linéaire basée sur des données issues de capteurs dans la chaussée et de temps de parcours historiques. Les résultats ont démontré que les temps de parcours issus des capteurs constituent une bonne prédiction à court terme (jusqu'à 20 min) tandis que les données historiques sont plus adaptées aux prédictions à long terme.

Ainsi, nous constatons que le type de données utilisé pour la prédiction a un impact sur l'horizon de prédiction (court, moyen ou long termes). Ce point sera d'ailleurs discuté dans le chapitre 7.

Méthodes non paramétriques.

Parmi les modèles non paramétriques pour la prédiction des temps de parcours, nous retrouvons les réseaux de neurones artificiels, les modèles de simulation, les modèles bayésiens, etc.

Les réseaux de neurones artificiels ont été appliqués à des problèmes de prédiction dans le secteur du transport en raison de leur capacité à traiter des données multidimensionnelles et leur pouvoir de généralisation. Les réseaux de neurones les plus populaires sont les Multi-Layer Perceptrons (MLPs) composés d'une couche d'entrée, d'une couche cachée et d'une couche de sortie, chacune d'elles contenant une ou plusieurs unités (neurones). Les unités dans la couche d'entrée sont connectées à celles de la couche cachée, qui à leur tour sont connectées à celles de la couche de sortie. Les poids de ces connexions sont ajustés durant la phase d'apprentissage. À titre d'exemple, à partir de données d'entrée (vitesse instantanée, flux, occupation) issues de capteurs installés sur des autoroutes aux environs de la ville de Londres, un MLP est utilisé dans [10] pour prévoir le flux de trafic. Dans [11], un MLP est utilisé pour prévoir le temps nécessaire pour traverser le pont Ambassadeur, l'un des plus fréquentés à la frontière canado-américaine. Une base de données de points GPS, enregistrés sur une année complète a été utilisée pour entraîner et tester le réseau. Un autre type de réseaux de neurones dont l'efficacité a été démontrée pour la prédiction sont les Recurrent Neural Networks (RNNs) en raison de leur capacité à traiter des données recueillies de façon séquentielle dans le temps. Généralement, le signal est envoyé de la couche cachée à elle-même ainsi qu'à la couche de sortie à l'instant t. Ce signal est traité avec le signal provenant de la couche d'entrée à l'instant $t+1$ afin de déterminer le nouvel état interne de la couche cachée. Cet état interne agit comme une mémoire qui permet de conserver les relations temporelles entre les données.

Dans [12], un RNN spécifique est utilisé, soit le Long Short-Term Memory (LSTM), qui est comparé à d'autres RNNs et méthodes statistiques. Le LSTM a alors fourni de meilleures performances en termes de précision et de stabilité.

Malgré les nombreux travaux portant sur la prédiction des temps de parcours et la diversité des données fournies en entrée aux méthodes utilisées, aucun travail n'a rapporté une démarche complète qui commence par le traitement de grands volumes de données pour en extraire de l'information selon des techniques de forage de données, jusqu'à la prédiction par le biais de techniques d'apprentissage automatique. Nous tentons, dans le chapitre 4 de fournir une solution complète pour produire des prédictions de vitesses de parcours sur un réseau routier d'une grande ville comme Montréal à partir de données GPS.

2.2 Optimisation Statique

Le second axe de recherche de cette thèse est celui des algorithmes d'optimisation permettant de résoudre un problème statique de livraison à domicile, où les vitesses sur les arcs du réseau varient dans le temps, mais où les variations sont connues à l'avance et ne changent pas. De façon générale, la figure 2.2, adaptée de [2], présente l'ensemble des méthodes de résolution développées pour les problèmes statiques de tournées de véhicules.

Il est désormais évident, dans un contexte urbain, que négliger les variations des vitesses et temps de parcours au cours de la journée, mène à des tournées de livraison de moindre qualité, voire non réalisables (c.f., vitesses aux heures de pointe et aux heures hors-pointe). Pour cette raison, des auteurs ont proposé des variantes des problèmes de tournées classiques où les temps de parcours ne sont pas fixés, mais varient en fonction du moment de la journée ($TDVRP$ pour Time-Dependent Vehicle Routing Problem). Toutefois, le graphe utilisé ne correspond pas au véritable réseau routier. Il s'agit en fait d'un **graphe client** où chacun des nœuds représente un client et où l'arc entre chaque paire de clients correspond à un chemin unique dans le réseau sous-jacent, calculé au préalable. Dans cette catégorie, le travail rapporté en 1992 dans [13] est le premier à avoir abordé un problème de tournées de véhicules dépendant du temps. Les auteurs ont divisé l'horizon temporel en intervalles et le temps de parcours sur chaque intervalle est modélisé à l'aide d'une fonction par paliers.

Cependant, ce modèle ne satisfait pas la propriété FIFO (First-In First-Out). Autrement dit, un véhicule peut partir plus tard qu'un autre d'un même point et arriver plus tôt à la même destination, même si les deux véhicules empruntent le même chemin. Dans [14], les auteurs ont modélisé les vitesses avec des fonctions par paliers, ou une journée est divisée en un nombre de périodes et une vitesse est associée à chacune d'elles. Ce modèle désigné par IGP (en raison du nom de ses auteurs) permet de respecter la propriété FIFO. De plus, pour un arc donné, la fonction de vitesse telle que définie par [14] peut aisément être convertie en une fonction des temps de parcours, linéaire par morceaux. Les auteurs divisent l'horizon de planification en trois périodes : périodes de pointe matinale et en après-midi et une troisième période à midi. Les résultats d'instances dérivées de celles de Solomon ont démontré les bénéfices de considérer un modèle qui varie en fonction du temps au lieu de temps de parcours constants.

Dans [15, 16], les auteurs ont résolu un $TDVRP$ avec fenêtres de temps souples en utilisant un algorithme génétique. Dans le premier de ces deux articles, le problème est dynamique puisque de nouveaux clients s'ajoutent au cours de la journée pendant l'exécution des tournées courantes. L'objectif est de minimiser les coûts fixes des véhicules, les coûts de transport et

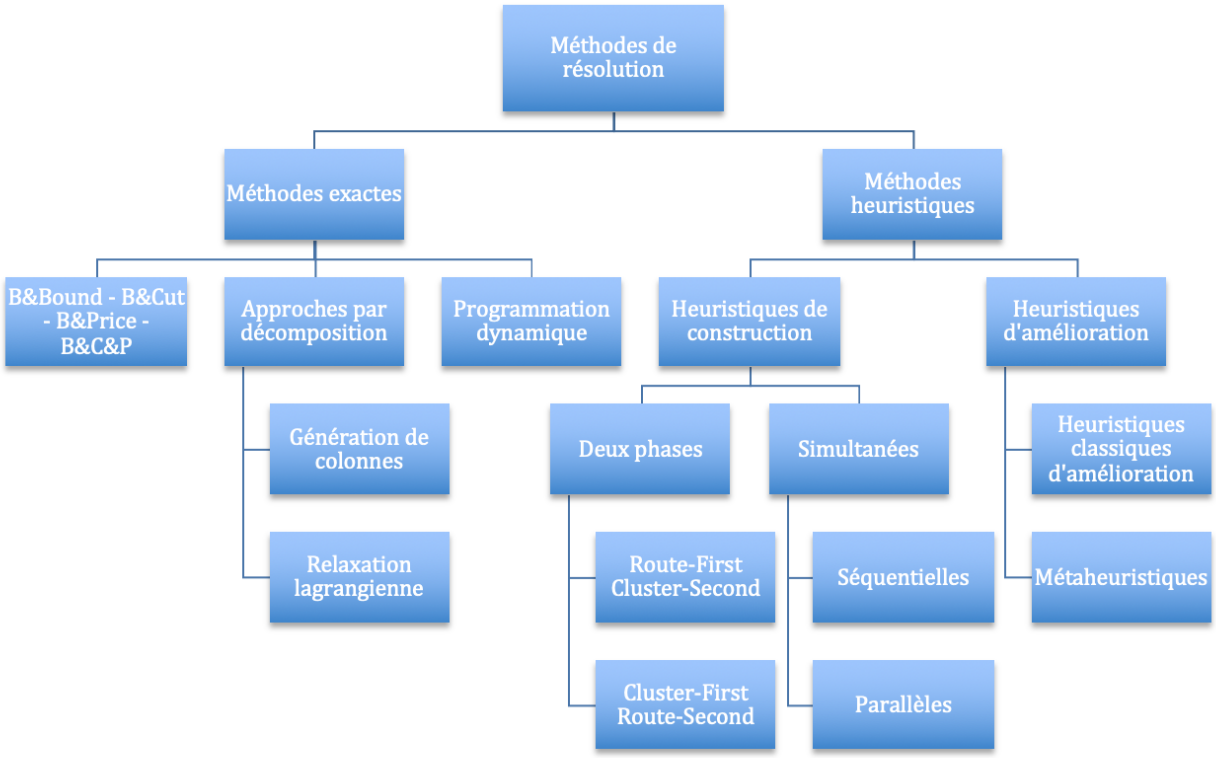


Figure 2.2 Classification des méthodes de résolution pour les problèmes de tournées de véhicules, adaptée de [2]

.

les coûts liés aux retards chez les clients. Les temps de parcours qui varient en fonction du moment de la journée sont modélisés par des fonctions continues garantissant la propriété FIFO.

Le problème $TDVRP$ a été résolu par [17] en utilisant deux méthodes, appliquées à un réseau routier réel en Italie : algorithme de colonies de fourmis et une recherche locale, tout en se basant sur le modèle IGP. Ensuite, les auteurs dans [18] ont proposé une méthode de construction itérative suivie d'une procédure d'amélioration, tel qu'illustré dans la catégorie des méthodes heuristiques de la figure 2.2.

Plus récemment, les auteurs dans [19] ont traité un $TDVRP$ avec fenêtres de temps souples et temps de parcours stochastiques. Ce problème est résolu grâce à une recherche tabou et une recherche adaptative à voisinage large.

Dans [94], les auteurs ont fourni un gabarit général pour implémenter divers algorithmes d'optimisation dans un contexte ou les données varient en fonction du temps, et ce, en présence d'un système de gestion de trafic. Les tests effectués sur la ville de Berlin ont démontré que considérer des temps de parcours constants mène à mal évaluer la durée totale requise d'environ 10%.

Ce n'est qu'en 2013 que les auteurs dans [20] proposent une méthode exacte pour le $TDVRP$ avec fenêtres de temps dont l'objectif est de minimiser la durée totale des routes. Pour cela, un algorithme de branch-and-price est développé et testé sur des instances de différentes tailles générées aléatoirement. Les instances de 25 nœuds ont été résolues à l'optimum de façon routinière tandis que seulement certaines instances de 100 nœuds ont pu être résolues.

Afin d'aborder des problèmes plus réalistes, certains chercheurs ont intégré de manière approximative le réseau routier à l'aide d'un **multigraphe-client**. On peut citer entre autres [21, 22] où les auteurs ont défini un mutigraphe avec des arcs parallèles entre deux nœuds (clients) qui représentent différents chemins qui peuvent être utilisés pour se rendre d'un client à l'autre. Dans [23], les auteurs définissent deux arcs entre chaque paire de clients, le premier représentant un état de congestion relativement faible et le second associé à une congestion routière sévère. Ainsi, selon le moment du départ, l'un des deux arcs domine l'autre. Les auteurs dans [25] définissent un $TDVRP$ où un sous-ensemble de chemins les plus courts sont calculés a priori pour chaque paire de clients, en utilisant différents moments de départ, à l'aide d'une variante de l'algorithme de Dijkstra qui tient compte de la dépendance au temps. Chaque chemin plus court est associé à une variable de décision dans un programme linéaire mixte en nombres entiers (MILP pour Mixed Integer Linear Program).

En 2014, les auteurs de [24] ont considéré un graphe qui contient deux catégories d'arcs :

les routes principales avec des vitesses qui dépendent du temps et de petites rues associées à une vitesse constante. En utilisant une procédure de type greedy randomized adaptive search procedure (GRASP) pour résoudre le problème, une amélioration de l'ordre de 12,5% dans les temps de parcours est observée par rapport aux solutions obtenues avec des vitesses constantes.

Le réseau routier peut également être représenté directement, donnant lieu au **graphe du réseau routier** où les intersections sont représentées par des nœuds, dont certains correspondent à des clients, et les segments routiers sont représentés par les arcs. Étant donné l'intérêt récent pour cette approche, la seule méthode exacte ayant considéré le graphe du réseau routier est [95]. Dans ce problème, les auteurs minimisent la distance totale parcourue et le temps n'intervient que sous forme de contraintes (fenêtres de temps). Ils considèrent trois approches :

- min-cost graph : le plus court chemin en temps entre deux clients est calculé; la fonction de temps de trajet correspondante est également calculée. Cependant, le problème est résolu sur ce graphe simple.
- min-time graph : le plus court chemin en temps est calculé en fonction de l'heure de départ (le chemin peut varier en fonction du moment de départ) ; les fonctions de temps de trajet et de coût de trajet sont maintenues. Cependant, le problème est résolu sur ce graphe simple.
- réseau routier : aucun pré-calcul n'est requis puisque tout chemin dans le réseau routier est accepté, tant que les contraintes de temps sont respectées.

Il est à noter que dans les trois cas, l'objectif demeure le même (minimiser la distance parcourue). La méthode exacte utilisée est un branch-and-price dont le problème esclave (pricing problem) est le *Time-Dependent Shortest Path Problem with Resource Constraints* résolu à l'aide d'un algorithme *time-dependent labeling* avec une recherche bi-directionnelle afin de réduire le nombre d'étiquettes générées.

Comme on peut le constater à partir de la revue de littérature relative aux problèmes statiques, aucune méthode heuristique n'a été rapportée à ce jour traitant du problème de confection de tournées de véhicules avec des temps de parcours qui varient en fonction du temps, considérant le graphe du réseau routier. Nous abordons ce problème, dans le chapitre 5, en utilisant une recherche tabou et en évaluant les solutions dans le voisinage en temps constant grâce aux techniques développées qui peuvent, par ailleurs, être réutilisées pour d'autres méthodes.

2.3 Optimisation Dynamique

Avec la collecte des données en temps réel, la capacité de tenir compte de ces données dans les algorithmes d'optimisation devient de plus en plus importante et nécessaire dans le secteur du transport. Dans ce contexte, avec l'augmentation de la puissance des ordinateurs, les récents développements en calcul parallèle et le développement des métaheuristiques, le domaine de l'optimisation dynamique ou optimisation en temps réel est apparu. Nous retrouvons en particulier la catégorie des problèmes dynamiques de tournées de véhicules.

De nombreuses revues de littératures ont abordé le problème dynamique de tournées de véhicules (DVRPs) :

- On retrouve notamment dans [106] une taxonomie de ces problèmes basée sur différents critères : (1) type du problème (2) contexte logistique (3) mode de transport (4) fonction objective (5) taille de la flotte (6) contraintes de temps (7) contraintes de capacité (8) capacité de rejeter des requêtes (9) source du dynamisme (10) source de la stochasticité, et (11) méthode de résolution.
- Dans [107], les auteurs mettent l'emphase sur les problèmes dynamiques avec des données déterministes et/ou stochastiques. Ces données peuvent être relatives à de nouvelles requêtes, à des temps de parcours dynamiques, etc.
- Dans [108], les problèmes dynamiques sont classifiés selon la qualité de l'information et son évolution. Les auteurs présentent différentes notions relatives au degré du dynamisme selon 1) le ratio entre le nombre de reqeuets dynamiques et le nombre total de requêtes, 2) la moyenne normalisée des mises à jour, et 3) le temps de réaction.

Dans ce qui suit, nous n'allons pas faire l'inventaire des problèmes dynamiques de tournées de véhicules en raison de leur nombre important, mais plutôt citer les travaux relatifs à l'aspect dynamique des vitesses et des temps de parcours. Ces derniers sont peu nombreux, c'est d'ailleurs ce qui a motivé la troisième contribution de cette thèse.

Requêtes dynamiques.

Des démarches purement réactives sont d'abord considérées pour résoudre les problèmes dynamiques. Il s'agit principalement de problèmes statiques définis et résolus à chaque nouvelle occurrence de requête (aussi désigné par [108] *optimisation périodique*). Parmi les travaux ayant eu recours à de telles procédures, nous retrouvons [111] ou de petits colis sont recueillis de différents endroits et ramenés au dépôt. Une recherche tabou avec une mémoire adaptative est modifiée pour tenir compte de nouvelles requêtes. Afin d'obtenir des temps

de réaction viables dans un contexte dynamique, les auteurs ont procédé à la parallélisation de l'implémentation.

On retrouve également [112], qui ont considéré l'Action de détourner un véhicule de sa position courante lorsqu'une demande arrive dans le voisinage.

Afin de dépasser l'aspect myope des approches réactives décrites ci-dessus, d'autres procédures reposent sur l'action d'accumuler un certain nombre de requêtes avant de procéder à la réoptimisation. Citons [113], ou l'horizon de planification est divisé en intervalles de durées égales et toutes les requêtes qui arrivent à l'intéreiur d'un intervalle de temps sont maintenues jusqu'à la fin de ce dernier. L'optimisation est opérée grâce à l'algorithme de colonies de fourmis. Une approche similaire est présentée dans [114].

Finalement, une nouvelle approche consiste à exploiter toute information stochastique relative à de futures requêtes. Cette information peut être prise en compte 1) implicitement, comme dans [115] ou des temps d'Attente sont introduits pour pouvoir répondre à d'éventuelles requetes, 2) ou explicitement, comme dans [116], ou différentes solutions sont envisagées selon les scénarios qui se présentent.

Temps de parcours dynamiques.

Dans [27], les auteurs présentent un système dynamique de routage, affectant les clients aux véhicules à mesure qu'ils arrivent dans le système. Ce dernier utilise des informations en ligne relatives aux temps de parcours, qui proviennent d'un centre de gestion du trafic. Dans [28], les auteurs ont utilisé un algorithme génétique pour résoudre un problème dynamique de tournées de véhicules qui incorpore de l'information en temps réel sur les temps de parcours. Une simulation du trafic est utilisée pour représenter les variations dans les conditions de circulation dues à d'éventuels incidents. Le modèle est appliqué à un réseau de 25 nœuds et 80 arcs et les résultats démontrent une diminution du coût total allant jusqu'à 3,7% en tenant compte des temps de parcours dynamiques. Les auteurs dans [29] ont étudié un problème dynamique de cueillette ou livraison avec fenêtres de temps, où certaines requêtes de clients arrivent de façon dynamique et où les temps de parcours varient aussi de façon dynamique. Le problème est formulé comme un MILP et est résolu exactement (pour un maximum de 10 clients). Afin de résoudre de plus grandes instances, une solution basée sur un algorithme génétique est proposée, ce qui a permis de produire des solutions à 7% de l'optimum. Cette méthode a également été appliquée à un réseau avec 382 nœuds et 1398 segments routiers.

La notion de tolérance pour les retards chez les clients est introduite dans [30]. Cette tolérance représente le retard maximal acceptable suite aux changements qui sont introduits dans les données du problème. Les auteurs décrivent un algorithme d'optimisation qui repose sur

une recherche locale et reproduisent l'aspect dynamique grâce à un simulateur d'événements discrets. Leur méthode est appliquée aux problèmes de tournées de véhicules avec fenêtres de temps de M.M. Solomon avec 100 nœuds et il est démontré qu'une tolérance faible aux retards permet d'améliorer considérablement la valeur de l'objectif. Ce travail a ensuite été étendu dans [31] en permettant de détourner les véhicules de leur destination courante (diversion), afin de répondre à de nouveaux clients qui apparaissent non loin d'un véhicule. Cette extension demande évidemment des outils permettant une communication directe entre le centre de répartition et les véhicules. Les résultats numériques ont démontré que la nouvelle approche permet d'améliorer encore davantage les solutions obtenues, par rapport au travail précédent. La même idée de tolérance est également reprise dans [32] où trois modèles différents sont proposés. Dans [33], la théorie des files d'attente est utilisée afin de modéliser les variations dynamiques des temps de parcours. Une comparaison avec d'autres approches est présentée, confirmant et quantifiant les bénéfices de ce modèle.

CHAPITRE 3 ORGANISATION DE LA THÈSE

Comme évoqué dans l'introduction, l'objectif de cette thèse est d'exploiter les méthodes de l'IA et de la recherche opérationnelle pour un problème de livraison à domicile dans un réseau urbain, où on retrouve des vitesses sur les arcs du réseau qui changent selon le moment de la journée, des fenêtres de temps pour le service aux clients et une capacité maximale pour les véhicules de livraison. À cet effet, après les deux premiers chapitres consacrés à l'introduction et à la revue de littérature sur les méthodes pertinentes en IA et en recherche opérationnelle, la suite de la thèse est organisée comme suit :

Le chapitre 4 est un article ayant été soumis à la revue scientifique *EURO Journal on Transportation and Logistics*, nous y présentons une méthodologie de traitement de données GPS, recueillies par des capteurs embarqués dans des véhicules de livraison, en faisant appel à des méthodes de forage de données et d'apprentissages automatiques supervisé et non supervisé. Le but est de prévoir des vitesses sur les arcs d'un réseau routier selon le moment de la journée à partir de données historiques. Dans le cadre de ce projet, nous avons collaboré avec un partenaire spécialisé dans le développement d'algorithmes de confection de tournées pour la livraison à domicile de biens achetés par des clients dans des commerces de détail. Ce travail revêt une dimension très pratique, puisque la méthodologie que nous avons développée a été appliquée au réseau routier de la grande région de Montréal.

Le chapitre 5 est un article ayant été soumis à la revue scientifique *Transportation Science*. Nous y développons une méthode de résolution pour le problème de livraison à domicile dans un contexte statique où toutes les données sont connues à l'avance et ne changent pas. En particulier, les vitesses diffèrent selon le moment de la journée, mais ces variations sont supposées connues. L'approche de résolution est une métaheuristique qui a déjà démontré son efficacité pour ce type de problème, soit la recherche tabou. On retrouve dans ce chapitre la description de techniques innovantes permettant l'évaluation en temps constant des solutions dans le voisinage de la solution courante, permettant ainsi de réduire considérablement les temps de calcul. Nous avons testé cette approche sur des instances rapportées dans la littérature et avons constaté que l'approche proposée produit des solutions de grande qualité en les comparant à des solutions optimales.

Le chapitre 6 est un article ayant été soumis à la revue scientifique *European Journal of Operational Research*. Nous considérons à nouveau le même problème, mais dans un contexte dynamique où les vitesses sur les arcs du réseau sont régulièrement mises à jour. Une procédure d'optimisation permettant de réagir à ces mises à jour dynamiques y est décrite

et comparée à une procédure non réactive. En amont de cette procédure d'optimisation, un simulateur à événements discrets permet de générer les perturbations aux vitesses, en tenant compte de leur probabilité d'occurrence selon le moment de la journée. Les résultats obtenus sur une portion du réseau montréalais sont très prometteurs, autant au niveau des temps de réaction qu'au niveau de la qualité des solutions proposées.

Dans une vision d'ensemble, le chapitre 4 a pour objectif de fournir, par le biais de prédictions, des données d'entrées au type de problèmes étudiés dans les chapitres 5 et 6. En outre, la plupart des méthodes définissent un problème de tournées de véhicules dynamique comme une série de problèmes statiques définis sur un horizon roulant. Ainsi, les méthodes de résolution de problèmes statiques servent généralement de base aux méthodes de résolution dynamiques. Ce lien est d'ailleurs représenté dans la figure 2.1.

Le manuscrit se termine par une discussion générale, suivie d'une conclusion, dans laquelle nous discutons et analysons les résultats obtenus dans l'ensemble de la thèse, précisons certaines limitations et introduisons des perspectives de recherche pour le futur.

CHAPITRE 4 ARTICLE 1: TRAVEL SPEED PREDICTION BASED ON LEARNING METHODS FOR HOME DELIVERY

Ce chapitre a été soumis pour publication sous forme d'article de revue scientifique : Gmira, M., Gendreau, M., Lodi, A., Potvin, J-Y. (2019). Travel speed prediction based on learning methods for home delivery. *EURO Journal on Transportation and Logistics,* under revision.

Abstract.

The travel time required to proceed from one location to another in a network is an important problem parameter in many urban transportation settings ranging from the planning of delivery routes in freight transportation to the determination of shortest itineraries in advanced traveler information systems. Accordingly, accurate travel time predictions are of foremost importance. In an urban environment, vehicle speed, and consequently travel time, can be highly variable due to congestion caused, for instance, by accidents or bad weather conditions. At another level, one also observes daily patterns (e.g., rush hours), weekly patterns (e.g., weekdays versus weekend), and seasonal patterns. Capturing these time-varying patterns when modeling travel speeds can provide an immediate benefit to commercial transportation companies that distribute goods, since it allows them to better optimize their routes and reduce their environmental footprint.

This paper presents the first part of a project aimed at optimizing time-dependent delivery routes in an urban setting. It focuses on the prediction of travel speeds using as input GPS traces of commercial vehicles accumulated over a significant period of time. The proposed forecasting framework is based on machine learning and data mining techniques: unsupervised learning, dimensionality reduction techniques and imputation methods, and deep learning models.

Keywords. traffic prediction, machine learning, data mining, dimensionality reduction.

4.1 Introduction

Travel time forecasting is a major issue in most urban transportation models, whether they deal with the transportation of goods or persons. Most variations in travel speeds, and hence in travel times, result from traffic congestion, which can be classified as recurrent (due to well-known patterns) and non-recurrent (due to accidents, construction, emergencies, special events and bad weather, among others). Accordingly, there is a need for accurate predic-

tions of travel times (or travel speeds) under recurrent and non-recurrent congestion. It is important to note that better forecasts may significantly help individuals and transportation companies operating in urban areas in planning their activities, and possibly lead to substantial reductions in actual travel times and greenhouse gas (GHG) emissions.

With the increasing amount of available data collected from probe vehicles, smartphone applications, and other location technologies, the challenge in travel time (or speed) forecasting is no longer related to the quantity of data, but rather to the modeling and extraction of useful information from such data. In this context, there is a great potential for the development of new models and algorithms based on real data that can accurately predict travel speeds at various times of the day for different groups of streets and roads.

The research presented in this paper is part of an integrated project that aims at capturing predictable patterns in travel speeds for commercial vehicles in an urban setting with the objective of optimizing time-dependent routes for a fleet of delivery vehicles. This paper focuses on making the best travel speed predictions possible using data collected from mobile location devices. This will be done with machine learning and data mining techniques, namely unsupervised learning to cluster data, dimensionality reduction techniques and imputation methods to fill missing values, and a deep learning model to produce the travel speed forecasts.

The data for this study come from a software development company located in Montreal that produces vehicle routing algorithms to plan the home delivery of large items (appliances, furniture) to customers. This partner has delivery routes with more than 2,500,000 delivery points, serviced by nearly 200,000 routes. Data are collected using Automatic Vehicle Location (AVL) systems where GPS receivers are usually interfaced with Global System for Mobile Communications (GSM) modems. The system records point locations as latitude-longitude pairs, instantaneous speed, date and time. Our challenge here was to develop techniques for the management of big data that would allow prediction algorithms to perform well.

The paper is organized as follows. We first review the scientific literature related to our work in Section 5.2. Then, our methodology for travel speed prediction is reported. The creation of a database of speed patterns from GPS traces is described in Section 4.3. Then, techniques to reduce the size of the database and cluster arcs into similarity classes are explained in Section 4.4. This is followed by a description of the neural network model used to predict travel speeds in Section 4.5. Computational results are finally reported in Section 4.6. The conclusion follows.

4.2 Literature review

Most urban traffic control systems rely on short-term traffic prediction and a huge literature has been published on this topic in the last decades due, in particular, to the advent of Intelligent Transportation Systems (ITS). Given that these systems are highly dependent on accurate traffic information, they must collect a large amount of data (locations, speeds and individual itineraries).

Travel speed prediction at a given time typically relies on historical travel speed values and a set of exogenous variables. Methods to predict traffic information are classified in [34] as 1) naive (i.e., without any model assumption), 2) parametric, 3) non-parametric and 4) a combination of the last two, called hybrid methods. The first three methods are described in the following.

4.2.1 Naive methods

Naive methods are by far the easiest to implement and to use because they do not require an underlying model. However, one main drawback is their lack of accuracy. In [34], naive methods are divided into instantaneous methods (based on data available at the instant the prediction is performed), historical methods (based on historical data) and mixed methods, where the latter combine characteristics of historical and instantaneous methods. As a baseline for comparison with other parametric methods, the work reported in [35] uses a simple hybrid method where the travel speed forecast is a function of the current traffic flow rate, as well as its historical average at a given time of the day and day of the week. In [36], the authors compare two methods for travel time prediction, one based on data available at the instant the prediction is performed and one based on historical data. Using the Relative Mean Error (RME) and the Root Mean Squared Error (RMSE), these two approaches showed similar performance, but were clearly outperformed by a more sophisticated approach called Support Vector Regression (see Section 4.2.3).

4.2.2 Parametric methods

Parametric methods use data to estimate the parameters of a model, whose structure is predetermined. The most basic model is linear regression, where the traffic variable V_t to be predicted at a given time t is a linear function of independent variables:

$$V_t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n \ . \tag{4.1}$$

Different techniques are available to estimate the parameters β_i, $i = 1, \ldots\ n$. Among parametric methods, we consider the Autoregressive Moving Average model (ARMA) and its Integrated variant (ARIMA), different smoothing techniques and the Kalman filter. They are presented below.

(a) Kalman Filter

The Kalman filter is a very popular short-term traffic flow prediction method. It allows the state variables to be updated continuously, which is very important in time-dependent contexts. Some works related to traffic state estimation are based on the original Kalman filter [37], as well as its extension for non-linear systems called the Extended Kalman Filter (EKF) [38]. The latter is particularly relevant since the travel times depend on traffic conditions that are highly non-linear and dynamic, changing over time and space. In [39], a freeway state estimator is obtained by solving a macroscopic traffic flow model with EKF. A new EKF based on online-learning is used in [40] to provide travel time information on freeways. Also, a dynamic traffic assignment model, which is a non-linear state-space model, is solved by applying three different extensions of the Kalman filter [41].

In [42], the authors use traffic data on California highways to predict travel times on arcs and estimate the arrival time at a destination. First, travel times on arcs are predicted by feeding the Kalman filter with historical data. Then, this prediction is corrected and updated with real time information using the Kalman filter's corrector-predictor form.

(b) ARMA

To predict short-term traffic characteristics such as speed, flow or travel time, time series models based on ARMA(p,q) (which is a combination of p autoregressive terms AR and q moving average terms MA) have been widely used. If we consider travel time prediction, the general formulation of ARMA(p,q) is:

$$T(t) - \sum_{i=1}^{p} \alpha_i T(t-i) = Z(t) + \sum_{j=1}^{q} \beta_j Z(t-j) \ , \tag{4.2}$$

where the travel time $T(t)$, given departure time t, is a linear function of the travel times at previous instants, $Z(t-1),\ \ldots .\ , Z(t-q)$ are noise variables and (α_i, β_j) are parameters.

ARIMA models generalize ARMA models for non-stationary time series. They rely on stochastic system theory since the processes are non-deterministic. In [43], the authors

use the Box-Jenkins approach [44] to develop a forecasting model based on ARIMA from data collected in urban streets (i.e., 1-minute traffic-volume on each street during peak periods). After comparing several ARIMA models, the one of order (0,1,1) yielded the best results in terms of traffic volume forecasts, where the values of 0, 1 and 1 refer to the order of the autoregressive model (number of time lags), number of differentiation steps (number of times the values in the past are subtracted) and moving-average terms, respectively. The authors in [35] compare a seasonal ARIMA model, called SARIMA, with a non-parametric regression model where the forecast generation method and neighbor selection criteria are heuristically improved. The tests showed that SARIMA performed better than the improved non-parametric regression.

In [45], the authors model traffic flow with SARIMA (1,0,1) and SARIMA(0,1,1) models. Another SARIMA model is reported in [46] to forecast traffic conditions over a short-term horizon, based on 15-minute traffic flow data. In this case, SARIMA outperformed the K-nearest neighbor method.

4.2.3 Non-parametric methods

Non-parametric methods include non-parametric regression and different types of neural networks. Non-parametric methods are also known as data-driven methods, because they need data to determine not only the parameters but also the model structure. Thus, the number and types of parameters are unknown a priori. The main advantage of these methods is that they do not require expertise in traffic theory, although they need a lot of data.

The most popular non-parametric methods for traffic prediction are the support vector machine, neural networks and non-parametric regression.

(a) Support Vector Machine

The Support Vector Machine (SVM) was first introduced in [47, 48] and used in many classification and regression studies. SVM is popular because it guarantees global minima, it deals quite well with corrupted data and works for complex non-linear systems. Support Vector Regression (SVR) is the application of SVM to time-series forecasting.

In [36], the authors analyze the application of SVR for travel time prediction. They used traffic data, obtained from an Intelligent Transportation System, over a five weeks period. The first four weeks correspond to the training set and the last week corresponds to the testing set. Using a Gaussian kernel function and a standard SVR implementa-

tion, their method improved the RME and RMSE when compared to instantaneous and historical travel time prediction methods. Due to its promising results, SVR has been used to predict traffic parameters such as traffic flow or travel time. However, the classical SVR, like the one used in [36], cannot be applied in real time because it requires a complete model training each time a new record (data) is added. There are also some variants of SVM for traffic prediction. In [49], the authors report a hybrid model called the chaos-wavelet analysis SVM that overcomes the need to choose a suitable kernel function. In the context of traffic flow prediction for a large-scale road network [50], SVM parameters are optimized by a parallel genetic algorithm, thus yielding a Genetic Algorithm-Support Vector Machine (referred to as GA-SVM).

(b) Neural networks

When considering data-driven methods, neural networks are among the best for traffic forecasting because of their ability to learn non-linear relationships among different features without any prior assumption.

The most widely used neural networks are called Multi-Layer Perceptrons (MLPs). They are typically made of an input layer, one hidden layer and an output layer, where each layer contains one or more units (neurons). The units in the input layer are connected to those in the hidden layer, while the units in the hidden layer are connected to those in the output layer. The weights on these connections are adjusted during the learning process using (input, target output) example pairs, so as to produce an appropriate mapping between the inputs and the target outputs. In [10], a MLP is used to predict traffic flow from input data (speed, flow, occupancy) collected by detection devices on a highway around the city of London. In that application, the MLP and a radial basis function network performed better than all ARIMA models considered. In [51], the authors exploit a genetic algorithm to fine tune the parameters and the number of hidden units in a MLP. Their model showed better generalization abilities when tested with new inputs. More recently, the authors in [52] report the performance of a MLP with 15 hidden units, trained with the Levenberg-Marquardt backpropagation algorithm. Based on different error performance indicators, the MLP showed good accuracy when predicting road speeds. In [11], a MLP is used to predict the time needed to cross the Ambassador bridge, one of the busiest bridges at the Canada-US border. A database of GPS records for a full year was used to train and test the neural network.

As indicated in [12], Recurrent Neural Networks (RNNs) are better suited for traffic forecasting tasks due to their ability to account for sequential time-dependent data.

Typically, the signal sent by the hidden layer to the output layer at some time t is also sent back to the hidden layer. This signal is processed with the input signal at time $t+1$ to determine the internal state of the hidden layer. This internal state acts as a memory and remembers useful time-dependent relationships among data.

A specific class of RNNs, called Long Short-Term memories (LSTM), is now widely used in the literature due to its proven ability to learn long-term relationships in the data. In [12], LSTM is compared to various RNNs and other statistical methods, namely: Elman neural network, non-linear autoregressive with exogenous inputs (NARX) neural network, Time-Delay Neural Network (TDNN), SVM, ARIMA and Kalman filter. The LSTM achieved the best performances in terms of accuracy and stability. The work in [53] proposes a LSTM model for short-term traffic flow prediction and compared it with Random Walk (RW), SVM, MLP and Stacked Autoencoder (SAE). The results showed the superiority of LSTM in terms of prediction accuracy, ability to memorize long-term historical data and generalization capabilities. In [54], LSTM and a neural network model made of Gated Recurrent Units (GRUs) [55] are applied to traffic data in California. The study showed that GRUs behave slightly better than LSTM, while both outperformed an ARIMA model in terms of Mean Squared Error (MSE) and Mean Absolute Error (MAE).

(c) Non-parametric regression

Another class of non-parametric methods is called Non-Parametric Regression (NPR). It is suitable for short-term traffic prediction since it can deal with the uncertainty in traffic flows. The objective of NPR is to estimate a regression function without relying on an explicit form, unlike the traditional parametric regression models (where the model parameters are estimated). The forecasting ability of NPR relies on a database of past observations. It applies a search procedure to find observations in this database that are similar to the current conditions. Then, it transfers these observations to the forecast function to estimate the future state of the system.

The k-Nearest Neighbor (k-NN) is a widespread class of non-parametric regression methods made of two components:

(i) Search procedure: the nearest neighbors (historical data most similar to the current input) are the inputs to the forecast function aimed at generating an estimate. The nearest neighbors are found using a similarity measure, which is usually based

on the Minkowski distance metric:

$$L_r = \left(\sum_{i=1}^{n} |p_i - q_i|^r \right)^{1/r} , \tag{4.3}$$

where n is the dimension of the state vector, p_i is the i^{th} element of the historical record currently considered, q_i is the i^{th} element of the current conditions, and r is a parameter with values between 1 and 2. The most common implementation uses a sequential search procedure. However, as the number of historical observations increases, the sequential search becomes very time consuming.

(ii) Forecast function: the most general approach to generate a prediction is to compute an average of the dependent variable values over the nearest neighbors. However, this approach ignores the information provided by the distance metric (i.e., past observations closer to the current input should have more impact on the forecast).

The authors in [56] were among the first to use NPR to estimate short-term traffic flows. Their work highlighted the importance of a large and representative dataset. NPR was then applied to estimate traffic volumes from two sites located in Northern Virginia Capital Beltway, based on five months of observations [57]. The results showed that the proposed method can generate more accurate predictions than the other tested approaches (including a neural network model). The work in [58] compares historical averages, time series, back-propagation neural networks and non-parametric regression models using a performance index that includes absolute error, error distribution, ease of model implementation and model portability. Overall, NPR proved to be better than the other models and was also easier to implement.

The interested reader is referred to [59] for a more exhaustive review on traffic forecasting. The following sections will now describe the various steps that we performed to produce travel speed predictions from our huge database.

4.3 Deriving speed patterns from GPS traces

Our industrial partner provided us with one year and a half of GPS data transmitted by mobile devices installed in delivery vehicles (extending over the years 2013, 2014 and 2015). These GPS points were first mapped to the underlying Montreal Metropolitan Community (MMC) network to generate daily speed patterns for each individual arc, where a daily speed pattern for a given arc is made of 96 average speeds taken over time intervals of 15 minutes.

In the following, the main issues related to the derivation of speed patterns from GPS traces are briefly discussed.

4.3.1 Data preparation

The record of each GPS point contains an identifier, a latitude-longitude pair, an instantaneous speed, a mobile identifier, a driver identifier, a date and a time stamp (Figure 4.1).

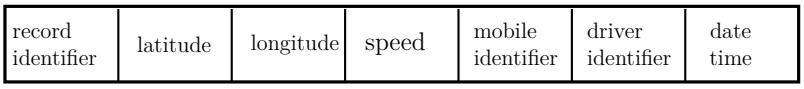


Figure 4.1 Record structure of a GPS point

The available data had first to be cleaned by deleting GPS points with aberrant speeds (values less than 0 km/h or greater than 150 km/h), aggregates of GPS points associated, for example, with parking stops for deliveries, etc. Then, the map-matching algorithm was applied to the remaining GPS points, as described below.

4.3.2 Map-matching algorithm

Due to the relatively low accuracy of GPS systems, assigning a GPS point to an arc of the underlying network is a difficult problem, particularly in dense urban road networks. Thus, a good map-matching algorithm is required. To this end, we used a recent algorithm reported in [60] which was slightly adapted to our context. The algorithm works as follows:

Step 1. Identification of trips. A total of 170 and 327 different vehicle and driver identifiers were found over all GPS points. Within a single day, it is possible to find one driver associated with one vehicle, one driver associated with two vehicles or more, and two drivers or more associated with one vehicle. Thus, there are clearly different trips within a day, where a trip corresponds to a vehicle/driver pair.

Step 2. For each trip, all GPS points associated with it are considered for assignment to arcs of the network. This is done in three main phases:

2.1 In the initialization phase, candidate arcs are those that are adjacent to the three nearest nodes of the first GPS point. A score is calculated for each arc based on their closeness to the GPS point, and the arc with the best score is selected. Then, a

confidence level is calculated for the selected arc to account for the uncertainty of that choice (due to inherent uncertainty in positioning sensors and digital maps). Here, the confidence level is based on the difference between the score of the selected arc and the second best score. Clearly, a larger difference implies a higher confidence level. If the confidence level is above a given threshold, the GPS point is assigned to the selected arc, otherwise it is skipped and the next GPS point is considered. Once a GPS point is assigned to the first arc, the same-arc phase starts.

2.2 In the same-arc phase, the next GPS point is assigned to the same arc than the previous one, unless some conditions are not satisfied anymore (e.g., when crossing an intersection). In the latter case, the algorithm switches to the next-arc phase.

2.3 In the next-arc phase, candidate arcs are those connected to the previously selected arc plus those that are adjacent to the three nearest nodes of the current GPS point. A score is calculated for each arc based on its closeness to the GPS point and the difference in direction between the arc and the line connecting the previous GPS point to the current one. Again, the arc with the best score is selected, its confidence level is calculated, and topological constraints are checked (e.g., arc not connected to the previous one, turn restrictions, etc.). Depending on the confidence level and satisfaction of the topological constraints, the current GPS point can either be skipped or assigned to the new arc. In the latter case, the algorithm returns to the same-arc phase. It should be noted that considering arcs that are close to the current GPS point, but not necessarily connected to the previous arc, allows the algorithm to account for discontinuities in GPS traces due to obstacles (e.g., tunnels, bridges).

The interested reader is referred to [60] for details about this algorithm.

When the assignment of GSP points to arcs is completed for each day, average travel speeds can be calculated over time slots of 15 minutes. Thus, a new database is obtained where each record has the structure shown in Figure 4.2. The next section will now explain how this database was exploited to fit our purposes.

4.4 Size reduction and clustering

Due to the size of our database, size reduction techniques were applied. After eliminating speed patterns and hours with too many missing data, a “prediction-after-classification”

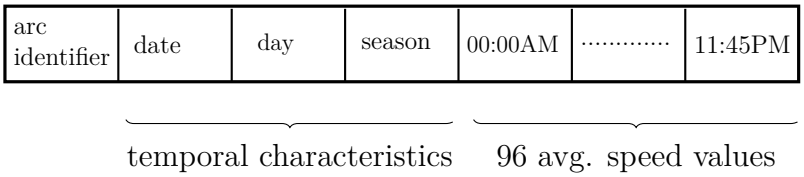


Figure 4.2 Database structure after geomatic analysis

approach was used to cluster arcs with similar speed patterns into classes before predicting travel speed values. This is explained in the following.

4.4.1 Database reduction

With 233,914 arcs and 515 days in the database, we have a total of 233,914 × 515 = 120,506,910 speed patterns. Originally, the database was constructed with time intervals of 15 minutes. That is, a speed pattern for a given arc on a given day is made of 96 average speed values taken over time intervals of 15 minutes, thus covering an horizon of 24 hours. Furthermore, the speed limit was stored when no observation was recorded within a 15-minute time interval.

An elimination procedure was first applied to get rid of speed patterns or time intervals with too few data, as described below.

(a) *Speed patterns.* To keep only average speed values based on real observations, the speed limit values were removed from the database. Given that a significant proportion of the resulting speed patterns now contained missing data, a speed pattern was automatically discarded when the proportion of real average speed values over the 96 time intervals was less than 5% (note that this threshold is often suggested in the literature [61]). In other words, a speed pattern with only 4 average speeds or less was eliminated. Through this process, we ended up with a total of 6,667,459 speed patterns. It should be noted that only 3,485 arcs still had at least one representative speed pattern in this database. The fact that the original GPS traces have been collected from delivery routes in particular sectors of an urban area explains this large reduction in the number of arcs and speed patterns. Finally, the 96 time intervals of 15 minutes of each remaining speed pattern were aggregated into 24 one-hour time intervals, to allow the calculation of average speed values based on more observations in each time interval, see Figures 4.3 and 4.4.

(b) *One-hour time intervals.* After reducing the number of speed patterns in the database,

as well as the number of average speed values stored in a pattern, we then examined more closely the one-hour time intervals.

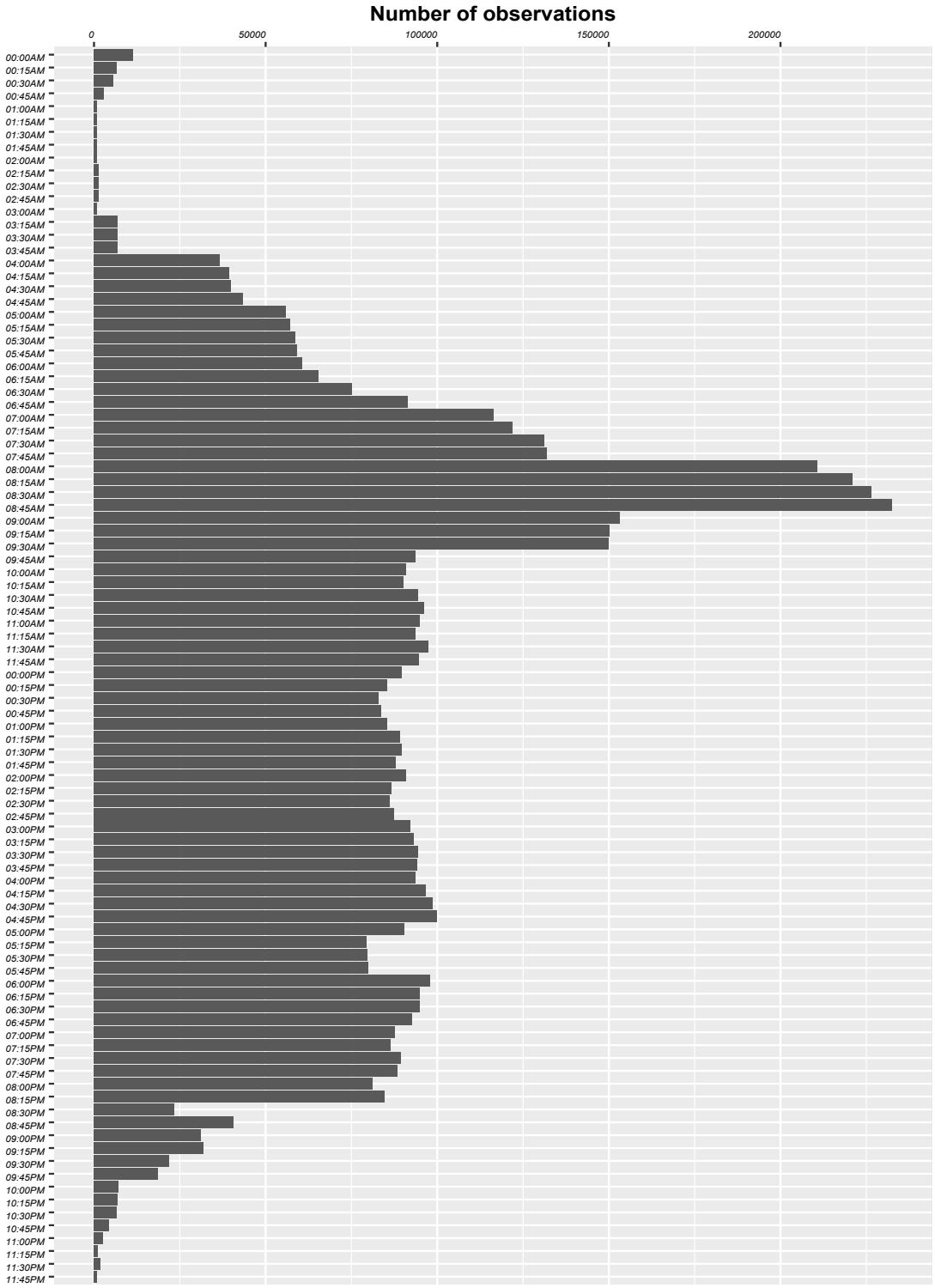


Figure 4.3 Number of observations per time interval of 15 minutes

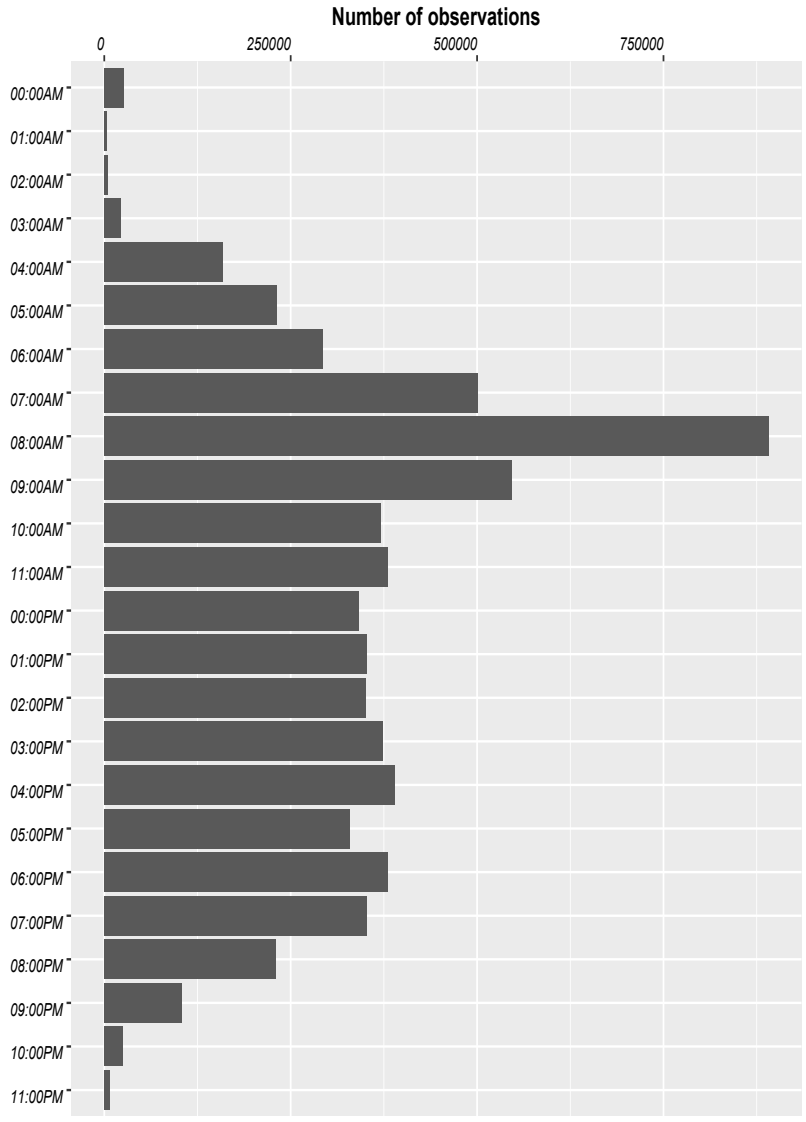


Figure 4.4 Number of observations per time interval of 1 hour

Clearly, there are intervals with no or very few observations over the entire database, like night hours (although some observations can be found in the database because the vehicles are sometimes moved during the night from one location to another). Accordingly, we discarded hours where the proportion of real average speed values over the 6,667,459 speed patterns in the database was under 5%. After this elimination process, the number of one-hour time intervals was reduced from 24 to 13. More precisely, only one-hour time intervals starting from 7:00 AM to 7:00 PM were kept in every speed pattern.

4.4.2 Clustering

When this step is reached, we have a database of 6,667,459 speed patterns, where each pattern is associated with a given arc and a given date. A speed pattern can be seen as a vector of 13 average speed values, one for each one-hour time interval starting from 7:00 AM to 7:00 PM. This number of speed patterns is still too large to be processed by a learning-based prediction algorithm, so we had to group arcs with similar patterns into a number of classes. For this purpose, an average speed pattern was calculated for each arc over all its corresponding speed patterns in the database. Since 3,485 arcs are represented in the database, this led to $N =$ 3,485 average speed patterns.

In the following, the clustering methods used to identify classes of arcs are described.

K-means

To cluster arcs based on their average speed pattern with the K-means algorithm, the distance between two patterns was calculated using the Euclidean metric (see, for example, [62]). At the end, K cluster centroids (classes) were obtained and the speed pattern of each arc was assigned to the closest centroid. Since the number of classes must be fixed in advance, the latter was purposely set to a large value, i.e., $K = 200$. Then, the output of the K-means algorithm was fed to a hierarchical clustering method to further reduce the number of classes (see Section 4.4.2).

The problem solved by the K-means algorithm is summarized below, where C_k stands for class k. The objective is to minimize the sum of the squares of the Euclidean distance between speed pattern x_i of arc i and its closest centroid μ_k, over all arcs. In this objective, the variables are the μ_k's.

$$Min \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2 \ , \tag{4.4}$$

where:

$$C_k = \{x_i : \|x_i - \mu_k\| = \min_{l=1,\ldots,K} \|x_i - \mu_l\|\} \ , \tag{4.5}$$

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i \ . \tag{4.6}$$

An iterative method known as Lloyd's algorithm was used to converge to a local minimum, given that solving the problem exactly is NP-hard. In this algorithm, starting from K randomly located centroids, the following two steps are performed repeatedly until convergence is observed: 1) Cluster assignment: construct the set of classes by assigning each arc to the cluster centroid that is closest to the arc's speed pattern and 2) Update centroids: update the centroid of each cluster by averaging over all speed patterns assigned to it.

Affinity propagation algorithm

The Affinity Propagation (AP) algorithm is a clustering procedure proposed in [63]. As opposed to K-means, every data point is considered as a potential centroid. Through the propagation of so-called affinity values among pairs of data points, which reflect the current affinity (or consent) of one data point to consider the other data point as its centroid, some data points accumulate evidence to be centroids. The reader will find in [63] the exact mathematical formulas that are used to guide the transmission of affinity values and the accumulation of evidence in data points. At the end, evidence is located only on a certain number of data points that are chosen as cluster centroids. Then, the set of classes is constructed by assigning each data point to its closest centroid. It should be noted that the centroids necessarily correspond to data points and that the number of classes does not need to be fixed in advance. That is, the number of classes will automatically emerge as the algorithm unfolds. The authors in [63] also show that AP approximately minimizes the sum of the squares of the Euclidean distance between each data point and its assigned centroid.

AP was applied using the centroids of the 200 classes produced by K-means as data points. At the end, these 200 classes were aggregated into 21 different classes, labeled from 0 to 20. Figures 4.5 and 4.6 show the number of arcs and observations, respectively, in each class.

It should be noted that it was not possible to apply AP directly to the $N = 3{,}485$ average speed patterns, which proved to be too large. This is the reason for the two-phase process,

where K-means is applied first to reduce the problem size, followed by AP in the second phase. We also tested another clustering algorithm, known as the Mean Shift Algorithm [64], but since it proved to be worse than AP, we will omit its description.

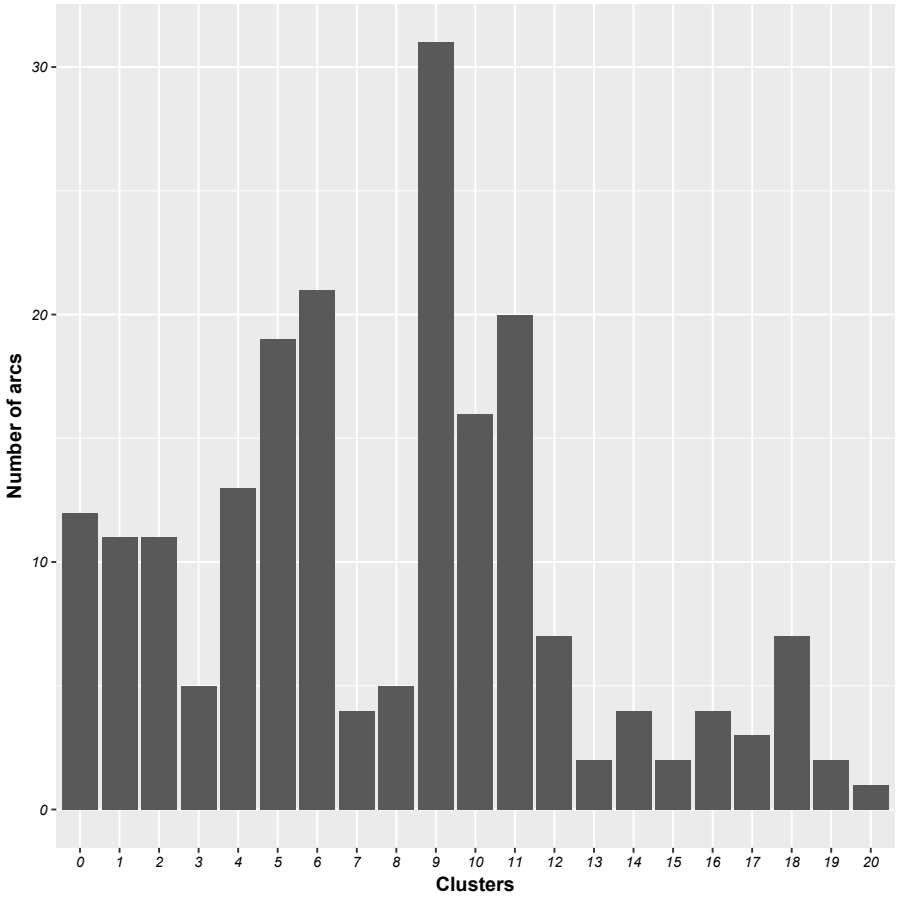


Figure 4.5 Number of arcs in each class generated by AP

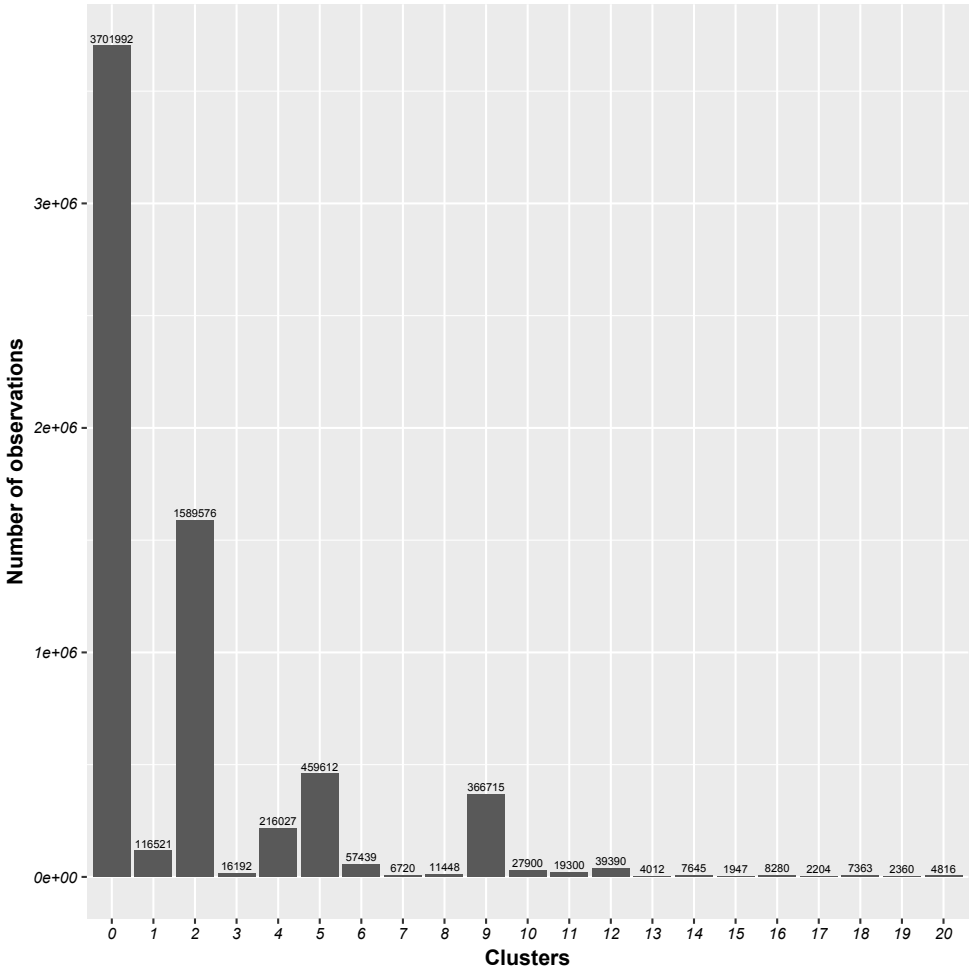


Figure 4.6 Number of observations in each class generated by AP

4.4.3 Evaluation metrics

After clustering the arcs into 21 classes, the quality of these classes was evaluated using two well-known metrics, namely the Silhouette coefficient and the Calinski-Harabasz score.

The Silhouette coefficient [65] associates a value between -1 and 1 with each data point. It can be interpreted as follows: if the coefficient is close to 1, then the data point is associated with the right cluster; if it is close to 0, then it lies between two clusters; and if it is close to -1, then the point is not associated with the right cluster. Assuming that a data point corresponds to the average speed pattern associated with an arc, this coefficient is calculated as follows:

– for every arc i, calculate the average distance between its speed pattern and the speed pattern of all other arcs in the same class; we call this value a_i.

– for every arc i, calculate the average distance between its speed pattern and the speed patterns of all arcs in the closest cluster; we call this value b_i.

– the silhouette coefficient s_i of arc i is then:

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)} \ . \tag{4.7}$$

At the end, the final value is the average of those s_i coefficients over all arcs.

The Calinski-Harabasz score [66] is another measure that provides a ratio between intra-class and inter-classes dispersion values. The clusters are better defined when the score is higher. The score is computed as follows:

$$CH^{(K)} = \frac{Tr(B^{(K)})}{Tr(W^{(K)})} * \frac{N-K}{K-1} \ , \tag{4.8}$$

where

$$W^{(K)} = \sum_{k=1}^{K} \sum_{x \in C^{(k)}} (x - c^{(k)})(x - c^{(k)})^{\intercal} \ , \tag{4.9}$$

$$B^{(K)} = \sum_{k=1}^{K} n^{(k)} (c^{(k)} - c)(c^{(k)} - c)^{\intercal} \ . \tag{4.10}$$

In these equations, a vector should be viewed as a column. Also, K is the number of clusters or classes, N is the number of speed patterns (arcs), $W^{(K)}$ is the intra-cluster dispersion matrix, $B^{(K)}$ is the inter-cluster dispersion matrix, $Tr(M)$ is the trace of matrix M, $C^{(k)}$ is the set, of cardinality $n^{(k)}$, of speed patterns (arcs) in class k, $c^{(k)}$ is the centroid of speed patterns in class k and c is the centroid of all speed patterns.

Note that each entry (i, j) in matrices $W^{(K)}$ and $B^{(K)}$ corresponds to:

$$W_{ij}^{(K)} = \sum_{k=1}^{K} \sum_{x \in C^{(k)}} (x_i - c_i^{(k)})(x_j - c_j^{(k)}) \ , \tag{4.11}$$

$$B_{ij}^{(K)} = \sum_{k=1}^{K} n^{(k)} (c_i^{(k)} - c_i)(c_j^{(k)} - c_j) \ . \tag{4.12}$$

When the Silhouette coefficient was evaluated on the 21 classes produced by AP, a value

of 0.85 was obtained. This is quite good, given that 1 is the best possible value (note that the coefficient for the clusters produced by the Mean Shift algorithm was equal to 0.67). Concerning the Calinski-Harabasz score, a value of 10.52 was obtained, which is to be compared with 3.39 for the Mean Shift algorithm.

By focusing on the four classes with the largest number of observations, namely classes 0, 2, 5 and 9, we observed that the average speeds of classes 0, 2 and 5 are very different, ranging from approximately 25 to 45 km/hour. The average speed of class 9 is similar to the one of class 2, but it does not evolve in the same way over the day. When we looked at more detailed data, we observed that the arcs of class 9 are not affected by the congestion observed during weekdays. That is, as opposed to classes 0, 2 and 5, there is no significant difference between the weekday average speeds and the weekend average speeds. Thus, the clustering algorithm was successful in identifying classes of arcs with different characteristics.

4.5 Speed prediction

This section describes the supervised neural network model for predicting travel speeds based on the classes generated by the AP clustering algorithm. Since a data missing issue emerges in this context, we will first explain how this problem is handled. Then, the neural network model will be described.

4.5.1 Missing data

Given that input vectors for the neural network model are obtained by averaging speeds over all arcs in a class produced by our clustering methodology, there is no missing data in the input (see Section 4.6.1). However, each target output vector corresponds to one of the 6,667,459 speed patterns in the database. Thus, it is likely for a target output to have one or more missing values.

To handle missing values, we must input plausible estimates drawn from an appropriate model. In this process, the following variables will be accounted for: day (Monday, Tuesday, ..., Saturday, Sunday), season (Spring, Summer, Fall, Winter), the arc's class label, and most importantly, the average speed in each time interval which corresponds to the variables with missing values. Different Multiple Imputation (MI) methods will be applied [67]. These methods generate multiple copies of an incomplete database and replace the missing values in each replicate with estimates drawn from some imputation method. An analysis is then performed on each complete database and a single MI estimate is calculated for each missing value by combining the estimates from the multiple complete databases [67,68]. The methods

considered here are: Multivariate Imputation via Chained Equation (MICE) [69], missForest (which relies on a Random Forest imputation algorithm [70]) and Amelia [71].

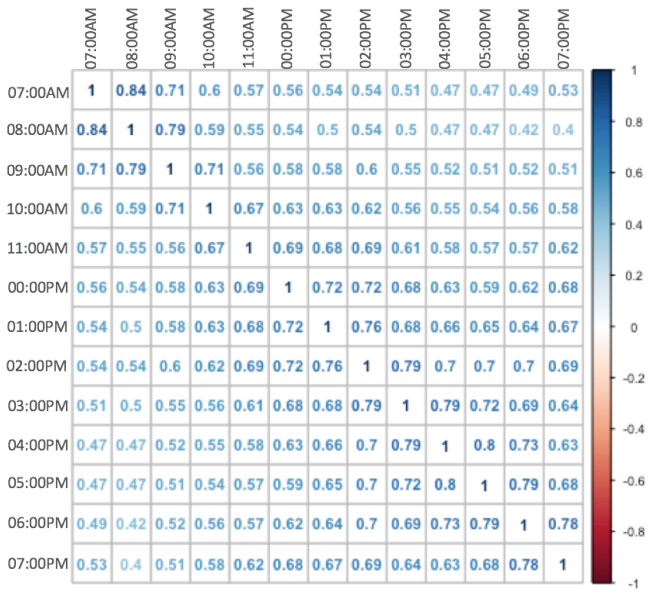


Figure 4.7 Correlation matrix between travel speeds

MICE

This algorithm can be described as follows:

1. Perform a mean imputation for every missing speed value by setting it to the average over observed speeds in the same time interval.
2. Select the time interval variable with the largest proportion of missing speed values.
3. Select the explanatory variables from those with a correlation greater than 0.5 with the selected time interval variable (see, e.g., the correlation matrix in Figure 4.7).
4. Perform linear regression.
5. Replace the missing speed values for the selected time interval with estimates obtained from the regression model. If this time interval is subsequently used as an explanatory variable in the regression model of other time interval variables, both observed and imputed values are used.

6. Repeat steps 2 to 4 for each remaining time interval with missing values (cycling through each variable stands for one iteration).

7. Repeat the entire procedure for a number of iterations to obtain multiple estimates.

At the end, the multiple estimates obtained over the iterations are averaged to obtain a single estimate for each missing value.

Random Forest

The Random Forest (RF) algorithm [72] is a machine learning technique that does not require the specification of a particular regression model. It has a built-in routine to handle missing values by weighting the observed values of a variable using a matrix of proximity values, where proximity is defined by the proportion of trees in which pairs of observations share a terminal node. It works as follows:

1. Replace missing values by the average over observed values in the same time interval.

2. Repeat until a stopping criterion is satisfied:

   (a) Using imputed values calculated so far, train a random forest.

   (b) Compute the proximity matrix.

   (c) Using proximity as a weight, impute missing values as the weighted average over observed values in the same time interval.

Typically, the algorithm stops when the difference between the new imputed values and the old ones increases for the first time. Note that, by averaging over multiple random trees, this method implicitly behaves according to a multiple imputation scheme. The RF algorithm was implemented using the randomForest R-package [73].

Amelia

Amelia imputes missing data based on different bootstrapped samples drawn from the database (bootstrapped data samples are large numbers of smaller samples of the same size that are repeatedly drawn, with replacement, from a single original sample, see [74]). It basically applies the Expectation Maximization (EM) method [75] to find the maximum likelihood estimates for the parameters of a distribution. Namely,

1. M samples are drawn from the original database.

2. M estimates of the mean and variance are calculated for each sample with EM.

3. Each estimate of the mean and variance is used to impute the missing data.

At the end, we have M different complete databases, where the missing values in each database have been filled with one of the M estimates of the mean and variance.

Each complete database produced by MICE, RF and Amelia is used to provide target outputs during the training and testing phases of our neural network model. The accuracy of the travel speed predictions made by the neural network will be compared for the three imputation methods considered.

4.5.2 LSTM

In this section, we briefly describe the supervised neural network model used to predict travel speeds, once the missing values in the database have been replaced by imputed ones (either using MICE, RF or Amelia).

The neural network model is a Long Short-Term Memory network (LSTM). This choice was motivated by the ability of the LSTM to handle sequential data and to capture time dependencies. First introduced in [76], this special type of Recurrent Neural Network (RNN) alleviates the vanishing gradient problem [77] (when the gradient of the error function becomes too small with respect to a given weight, the latter cannot change anymore). It is made of an input layer, a variable number of hidden layers and an output layer. Each hidden layer is made of memory cells that store useful information from past input data. Memory cells in a given hidden layer send signals at time t to the memory cells in the next hidden layer (or the units in the output layer, if last) but also to themselves. This recurrent signal is used by the memory cells to determine their internal state at time $t + 1$. Thus, there are connection weight matrices from the input to the first hidden layer, from each hidden layer to itself and to the next hidden layer and from the last hidden layer to itself and to the output layer. Furthermore, memory cells in a given layer have three gates: one for the signal sent from the previous layer, one for the signal sent by the memory cells to themselves and one for the signal sent by the memory cells to the next layer. Gates can be seen as filters that regulate the signals by allowing some parts of it to be blocked (or forgotten). Like the weight matrices mentioned above, the gates have weights that are updated during the learning process. The interested reader will find more details about the LSTM network model in [78].

In the next section, computational results obtained with LSTM and comparisons with alternative approaches are reported.

4.6 Computational study

In this section, we first define the input and output vectors of the LSTM. Then, we describe the fine tuning of the LSTM hyperparameters. Finally, LSTM results are reported.

Our LSTM was implemented in Python 3.5. The hyperparameter tuning experiment was performed on a Dell R630 server with two Intel Xeon E5-2650V4 of 12 cores each (plus hyperthreading) and 256GB of memory. The server also has 4 NVIDIA Titan XP GPUs with 3840 CUDA cores and 12GB of memory. However, our code was limited to only 4 cores and one GPU from the server. To obtain more computation power, the final LSTM results reported in Sections 4.6.3 and 4.6.4 were obtained on the Cedar cluster of Compute Canada. We requested 6 cores with Intel E5-2650v4 processors, 32GB of RAM and 1 NVIDIA P100 GPU with 12GB of memory.

4.6.1 Input and output vectors

The input vector for the neural network was first designed as illustrated in Figure 4.8. Each vector is associated with a class of arcs produced by AP and is made of: the class label, the day (Monday, Tuesday, ..., Saturday, Sunday), the season (Spring, Summer, Fall, Winter) and 13 average speeds over all arcs in the corresponding class, that is, one speed value for each one-hour time interval starting from 7:00 AM to 7:00 PM. The target output vector corresponds to a speed pattern among the 6,667,459 available speed patterns, where the missing values are filled with one of the three imputation methods of Section 4.5.1. Obviously, the target output vector must come from an arc of the same class, and for the same season and day than the vector provided in input. We should also note that the speed values in the patterns were normalized using the scikit-learn object [79].

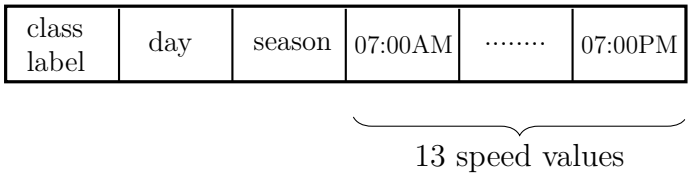


Figure 4.8 Input vector I

Unfortunately, the results obtained with this approach were unsatisfactory. To better exploit the capabilities of LSTM to handle sequential data, we turned to input vector II shown in Figure 4.9. Here, input vector I with 13 speed values is transformed into 13 input vectors II, each with a single speed value. That is, rather than providing at once the whole speed

pattern for one-hour time intervals starting from 7:00 AM to 7:00 PM, a sequence of 13 input vectors from 7:00 AM to 7:00 PM is provided, where each input vector contains a single speed value. The target output vector is modified accordingly and also contains a single speed value taken from an arc of the same class, and for the same season, day and hour than the vector provided in input.

Figure 4.9 Input vector II

4.6.2 Hyperparameter tuning

Our database of 6,667,459 speed patterns was divided into a training set (80% of the total) and a testing set (20% of the total), where the latter is made of the most recent observations. Apart from the connection weights, which are adjusted through learning, a neural network model also relies on a number of other parameters (where parameter is taken in a broad sense) that must be set before learning takes place. The following were considered:

- Number of hidden layers;
- Number of units in each hidden layer;
- Batch size: Number of training examples provided to the neural network before updating the connection weights;
- Training epochs: Number of passes through the set of training examples;
- Learning algorithm: Algorithm used during the training phase to adjust the weights;
- Weight initialization: Method used to set the initial weights, see [80] for more details;
- Activation function: Function used to compute the internal state of a unit from the signal it receives.

A good parameter setting for a neural network has a huge impact on its results, as discussed in [81]. To determine an appropriate combination of the above parameters, different hyperparameter optimization strategies can be used [82], in particular grid search and random

search. In grid search, a systematic evaluation of all possible combinations of parameter values is performed. Random search is much less computationally expensive, given that only a sample of all possible combinations is considered (100 combinations, in our case). Due to the curse of dimensionality, the superiority of random search over grid search is known for high-dimensional parameter spaces [83]. Since the number of parameters considered in our study is neither large nor small, both search methods were applied to our LSTM. The values tested for each parameter and the final setting obtained by each method are shown in Table 4.1. Note that the activation function differs depending on the hidden layer considered: the sigmoid function is used for the first hidden layer while the hyperbolic tangent (tanh) is used for the second and third hidden layers.

Table 4.1 Hyperparameter tuning using grid search and random search

Hyperparameters	**Values**	**Grid search**	**Random search**
Hidden layers	1,2,3,4,5	3	3
Units in each hidden layer	1,5,10,15,20,25,30,35,40,45,50	20	20
Batch size	10,20,30,40,50,80,100	20	20
Training epochs	5,10,20,30,40,50	10	10
Learning algorithm	SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam	SGD	Adam
Activation function	softmax, softplus, softsign, relu, tanh, sigmoid, hard sigmoid, linear	sigmoid tanh tanh	sigmoid tanh tanh
Weight initialization	uniform, lecun uniform, normal, zero, glorot normal, glorot uniform, he normal, he uniform	normal	normal

Overall, the only difference between the two hyperparameter optimization methods is the learning algorithm. Grid search suggests Stochastic Gradient Descent (SGD) while random search suggests Adam. The latter, introduced by [84], is particularly appropriate for complex neural network structures and large datasets, as detailed in [85]. To get a better idea, we applied the various tested learning algorithms (see Table 4.1) with the other hyperparameters fixed at their best value on a subset of the database. The results obtained with the RMSE metric are shown in Figure 4.10. Based on this preliminary experiment, we decided to go with Adam.

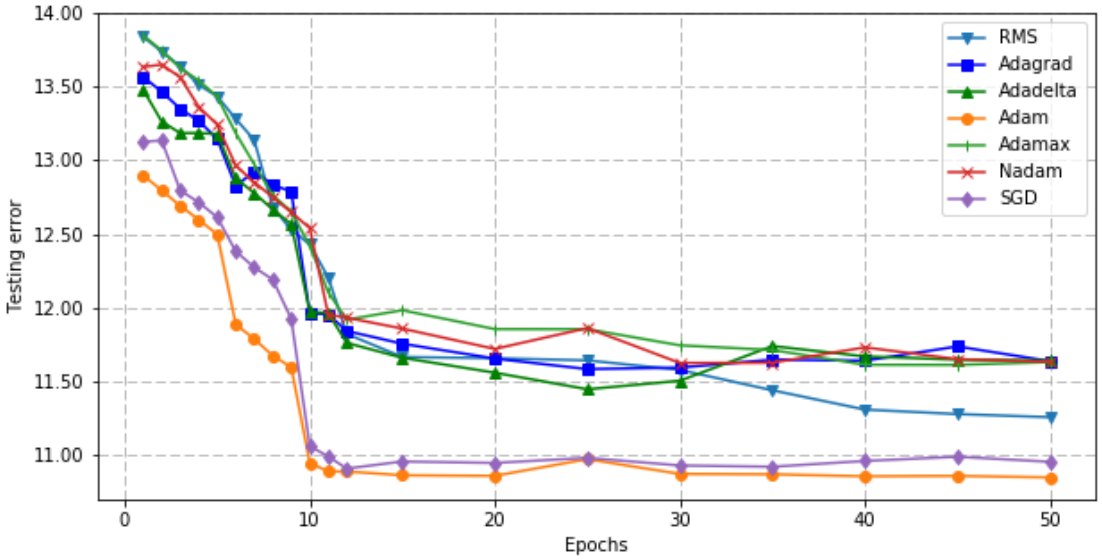


Figure 4.10 Initial comparison of different learning algorithms

4.6.3 LSTM results

Here, we measure the accuracy of the travel speed predictions produced by our LSTM with three hidden layers. We also compare the results obtained with the three imputation methods of Section 4.5.1 to fill the missing values.

The root mean squared error and the mean absolute error are used to measure the accuracy, where:

$$RMSE = \sqrt{\frac{1}{L}\sum_{l=1}^{L}(y_l - \hat{y}_l)^2}\ , \tag{4.13}$$

$$MAE = \frac{1}{L}\sum_{l=1}^{L}|y_l - \hat{y}_l|\ . \tag{4.14}$$

In these equations, L is the number of (input, target output) pairs in the training or testing set, where the target output corresponds to an observed speed pattern. In pair l, y_l stands for the observed speed pattern and $\hat{y}_l$ for the speed pattern produced by the neural network for input vector l. RMSE and MAE are two error functions typically used to evaluate how close the output vectors produced by the neural network are to the target outputs. Due to

the square in the RMSE formula, large errors have much more impact than small ones. On the other hand, MAE measures the relative error and is more robust to outliers since there is no square in its formula.

Table 4.2 reports the prediction errors of the trained LSTM on the testing set, based on the RMSE and MAE metrics, using the three different imputation methods and a variable number of imputations (i.e., 5, 10, 15, 20). Note that the authors in [86] show that 3 to 5 imputations yield good results. But, more recently, the authors in [87, 88] proposed 20 imputations. Thus, we chose to vary this number between 5 and 20.

Table 4.2 RMSE and MAE metrics with different imputation methods

Method	# Imput.	MAE	RMSE	Running time (min)
MICE	5	13.71	12.91	429
	10	**12.84**	**10.84**	450
	15	12.52	10.32	1117
	20	12.39	10.09	1900
RF	5	13.84	13.28	512
	10	13.02	11.38	649
	15	12.94	11.08	1640
	20	12.68	11.03	2000
AMELIA	5	17.91	17.53	550
	10	17.63	16.97	580
	15	16.76	16.10	940
	20	16.49	16.07	1500

The results show that MICE produces the best results. Although the two error metrics keep improving with the number of imputations, most of the improvement occurs between 5 and 10 imputations. Concerning the computing times, a substantial increase is observed between 10 and 15 imputations for all methods. Overall, MICE is the least computationally expensive for 5, 10 and 15 imputations. The marginal improvement obtained with 15 and 20 imputations probably does not justify the additional computational cost. Thus, MICE-10 seems to be a good compromise.

Figure 4.11 summarizes the evolution of the MAE and RMSE metrics for LSTM with MICE-10 over a number of training epochs. We can see that RMSE drops sooner than MAE and then keeps improving at about the same pace than MAE. Figure 4.12 then illustrates the differences between the observed speeds and the speeds predicted by the trained LSTM on the training and testing (input, target output) pairs on a sample of observations. This figure

shows that, in most cases, the predicted speed values follow closely the observed ones.

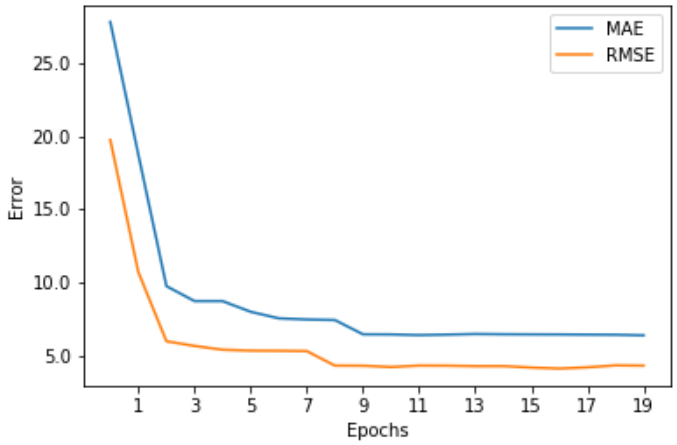


Figure 4.11 Evolution of MAE and RMSE during the training phase

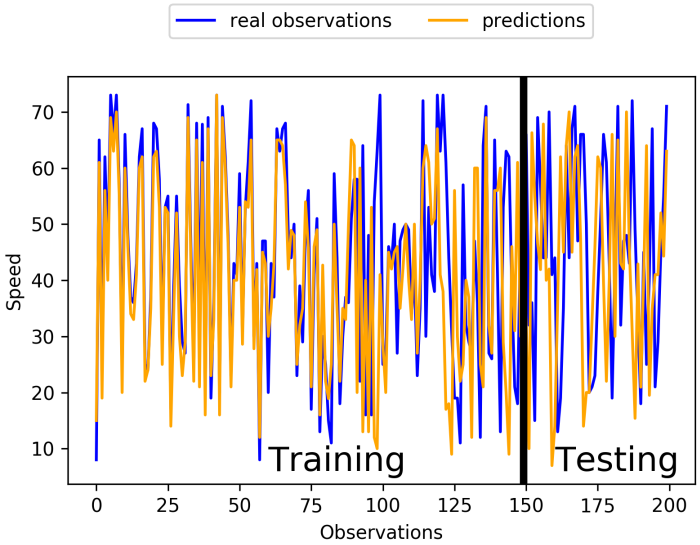


Figure 4.12 Comparison between observed and predicted speed values

4.6.4 Comparison with other models

In this section, LSTM is compared with two alternative approaches based on a trivial average method and a Multi-Layer Perceptron (MLP) trained with the Levenberg-Marquardt algorithm [89]. We tested MLP models with one to three hidden layers, using a variable number of units in the hidden layers. At the end, the error metrics of the various models were quite

close, although a model with three hidden layers and 30 units in each hidden layer proved to be the best. Thus, we only report the results for the best MLP model in Tables 4.3 and 4.4.

The average method is quite simple: if we consider an input vector of type I (II) for a given class, day and season (and hour), the output is the average speed pattern (value) over speed patterns (values) of arcs in the same class, for the same day and season (and hour). The results obtained with the LSTM, MLP and average method are reported in Tables 4.3 and 4.4 using the RMSE and MAE metrics, respectively. The results for the two input structures are also reported to show the superiority of input vector II over input vector I, see Section 4.6.1. Overall, LSTM clearly provides a better prediction accuracy than the two other models for both the RMSE and MAE metrics.

Table 4.3 Comparison of alternative models: RMSE metric

Model	Input vector I	Input vector II
LSTM	**17.22**	**10.84**
MLP	21.92	15.17
Average	38.04	32.04

Table 4.4 Comparison of alternative models: MAE metric

Model	Input vector I	Input vector II
LSTM	**22.86**	**12.84**
MLP	29.57	17.40
Average	42.62	36.98

4.7 Conclusion

In this work, a “prediction-after-classification" approach was proposed, starting from a large database of GPS traces collected from mobile devices installed inside delivery vehicles. We used dimensionality reduction and unsupervised learning techniques to extract similarity classes of arcs to be used during the following supervised prediction phase.

Because supervised learning algorithms are sensitive to missing values, different multiple imputation methods were applied to obtain a complete database. The prediction problem was addressed with a LSTM neural network whose parameters were adjusted with random search. The LSTM was then compared to two alternative models and prove to be largely superior.

A natural extension of the work reported in this paper would be to include real-time data in the prediction process. The ultimate goal is to integrate the current predictor into a vehicle routing optimization procedure to produce more efficient delivery routes.

Acknowledgments. Financial support was provided by the Natural Sciences and Engineering Research Council of Canada through the Canada Excellence Research Chair in Data Science for Real-Time Decision-Making. Also, computing facilities were made available to us by Compute Canada. This support is gratefully acknowledged. Finally, we wish to thank Prof. Leandro C. Coelho and his team for their contribution to the analysis of the GPS data.

CHAPITRE 5 ARTICLE 2 : TABU SEARCH FOR THE TIME-DEPENDENT VEHICLE ROUTING PROBLEM WITH TIME WINDOWS ON A ROAD NETWORK

Ce chapitre a été soumis pour publication sous forme d'article de revue scientifique : Gmira, M., Gendreau, M., Lodi, A., Potvin, J-Y. (2019). Tabu search for the time-dependent vehicle routing problem with time windows on a road network. *Transportation Science,* submitted for publication.

Abstract.

Travel times inside cities often vary quite a lot during a day and this may have a significant impact on the duration of routes. Several authors in the past have suggested time-dependent versions of the most commonly encountered vehicle routing problems. In these papers, however, time-dependent variations are usually defined with respect to customer-based graphs. Because of that, one of the major impacts of travel time variations is missed: in an urban environment, not only do travel times change, but also the paths used to travel from one customer to another. In fact, during a day, several different paths may be used at different points in time. To account for this, one must therefore work directly with road-based graph and consider travel time (or travel speed) variations on an arc by arc basis, if realistic delivery routes are to be determined.

In this paper, we propose a solution approach to solve a time-dependent vehicle routing problem with time windows in which travel speeds are defined on the road network itself. This solution approach involves a tabu search heuristic that considers different shortest paths between any two customers at different times of the day. A major contribution of this work is the development of techniques to evaluate the feasibility as well as the approximate cost of a solution in constant time, which allows the overall solution approach to handle instances with up to 200 nodes and 580 arcs in very reasonable computing times. The performance of the solution approach is assessed by comparing it to an exact method on a set of benchmark instances. The numerical results show that solutions of very high quality are produced.

Keywords. Vehicle routing, time windows, time-dependent travel times, road network, metaheuristics, tabu search.

5.1 Introduction

For more than 25 years now, it has been reported that neglecting variations in travel times in cities due, for example, to congestion during peak hours in the early morning and late afternoon, can lead to inefficient or even sometimes infeasible delivery routes. Several authors in the past have suggested time-dependent versions of the most commonly encountered vehicle routing problems. In these papers, however, time-dependent variations are usually defined with respect to customer-based graphs where the nodes are customers and an arc between two customers corresponds to a fixed shortest path previously calculated in the underlying road network.

The Time-Dependent Vehicle Routing Problem with Time Windows on a Road Network ($TDVRPTW_{RN}$) is aimed at producing more realistic routes by taking into account the time of the day to compute the shortest path (in time) in a road network to travel from one customer to the next. That is, not only do we observe different travel times during the day to go from one customer to the next, but even the paths used are different. It is clear that many applications may benefit from efficient problem-solving methodologies for the $TDVRPTW_{RN}$, like home delivery services.

Considering different paths to travel from one customer to the next in the road network, as well as accounting for the time-dependent travel time on each one of those paths, make the problem much more complex (which may explain the few works reported in the literature that address this problem). In this work, we propose a metaheuristic approach based on tabu search to efficiently solve benchmark instances of the $TDVRPTW_{RN}$ on road network graphs with up to 200 nodes and 580 arcs. This is achieved first by testing the feasibility of a neighbor solution in constant time and, second, by approximately evaluating the solution, again in constant time. The computational results demonstrate that near-optimal solutions are produced with this approach.

In the following, Section 5.2 provides a literature review about 1) time-dependent VRPs on customer-based graphs, 2) time-dependent VRPs on customer-based multigraphs and 3) time-dependent VRPs on road network graphs. Section 5.3 provides a formal description of our problem. Section 5.4 describes the problem-solving approach for the Time-Dependent Shortest Path Problem ($TDSPP$) on a road network, while Section 5.5 focuses on the tabu search heuristic. Section 5.6 describes the techniques used to assess the feasibility and evaluate solutions in constant time. The computational experiments and results are presented in Section 5.7. Finally, we conclude and state future research directions in Section 5.8.

5.2 Literature review

Many papers, as pointed out in [90–92], consider that travel times between two customers are constant over the day, which is rather unrealistic in practice. This assumption hugely impacts solution quality and may result in sub-optimal or even infeasible solutions, as shown in [14, 17]. In the following, we thus review the various approaches proposed in the literature to solve VRPs with time-dependent travel times.

5.2.1 Time-Dependent VRP on customer-based graphs

In this section, we consider exact and heuristic methods for solving $TDVRPs$, where the shortest path between two customers is precomputed and does not change over time, although the time to travel along that path depends on the departure time. In this case, the problem is defined on a customer-based graph, where an arc between two customers stands for a fixed path in the underlying road network. A good overview of this class of problems can be found in [93].

Heuristics

The work in [13] was the first, in 1992, to deal with a $TDVRP$ without time windows. The time horizon was divided into a few intervals and the travel time on each arc was modeled as a stepwise function with a different travel time associated with each interval. Unfortunately, this model does not satisfy the First-In-First-Out (FIFO) property. That is, a vehicle can depart later than another vehicle and arrive earlier at destination, even if the same path is followed by the two vehicles. This situation occurs, for example, if one vehicle waits just a little before departing to catch a shorter travel time associated with the next interval.

The authors in [14] were the first to consider a $TDVRPTW$, in this case with soft time windows. As opposed to [13], stepwise speed functions are proposed, like the one illustrated in Figure 5.1(a), where the day is divided into a number of time periods and a speed is associated with each time period. This model, referred to as IGP (due to the initials of the authors), satisfies the FIFO property since a vehicle can only arrive later at destination if it departs later. For a given arc, a stepwise speed function can easily be translated into a corresponding piecewise linear travel time function, as shown in Figure 5.1(b). In [14], the time horizon is divided into three time periods, one each for the early morning and late afternoon peak hours and one for mid-day. Computational results are reported on instances derived from Solomon's classical VRPs with time windows (VRPTW). The benefits of time-dependency are demonstrated by comparing the reported solutions with those obtained with

constant travel times.

In [15, 16], the authors solve a $TDVRP$ with soft time windows using a genetic algorithm. In the first work the problem is dynamic, that is, new customer nodes occur during the day and must be integrated in the solution while the current routes are being executed. The objective to be minimized includes fixed costs for the vehicles, routing costs and user inconvenience costs (lateness). In this work, time dependency is modeled with continuous travel time functions that must satisfy different assumptions to guarantee the FIFO property.

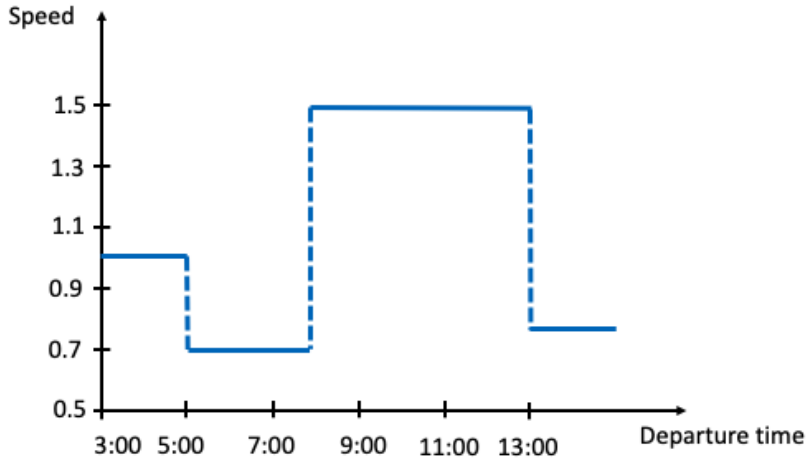
The authors in [17] address the $TDVRP$ with hard time windows using a hierarchy of two artificial ant colonies and a local search method, using the IGP time-dependency model. Their method is applied to instances derived from Solomon's VRPTW instances, as well as a real network in Italy. The time-dependent problem with soft and hard time windows is considered in [18], where a fast iterative route construction and improvement method is proposed, still based on the IGP time-dependency model.

In [19], the authors deal with a $TDVRP$ with soft time windows and stochastic travel times. A tabu search and an adaptive large neighborhood search are proposed to optimize both service efficiency and reliability.

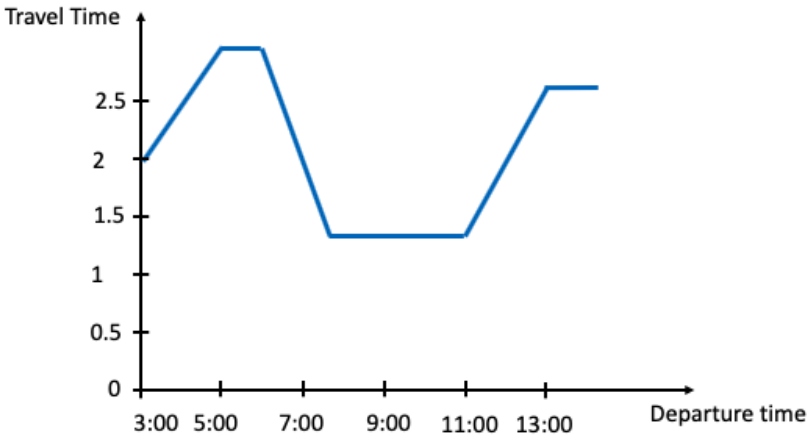
In the context of a traffic information system, the authors in [94] provide a general framework to implement time-dependency in various algorithms for the TDVRPTW. Computational tests based on real traffic data from the city of Berlin show that using constant average travel times underestimates the total travel time by approximately 10%.

Exact method

It is only in 2013 that an exact method to solve a $TDVRP$ with time windows was proposed [20]. In this paper, the IGP time-dependency model is used and the objective is to minimize the total duration of the routes. A branch-and-price algorithm is developed where the master problem is a set-partitioning problem and the pricing problem is a time-dependent shortest path problem with resource constraints, which is solved by a tailored labeling algorithm. The latter is aimed at identifying new feasible routes (columns) of negative reduced cost for the master problem in a typical column generation scheme. When solving the pricing problem, a label is associated with each customer to store the service completion time, the route duration and its reduced cost. Dominance criteria are introduced to discard labels that cannot lead to profitable routes while considering, at the same time, different departure times from the depot. Heuristics are also used to quickly find routes or columns of negative reduced cost. Experiments conducted on randomly generated instances of different sizes show that



(a) Speed function



(b) Travel time function

Figure 5.1 Piecewise linear travel time function derived from stepwise speed function of an arc of length 2

instances with 25 vertices can be routinely solved to the optimum. Even a few instances with 100 vertices were successfully addressed. As far as we know, no other authors have reported an exact method for tackling $TDVRPs$ with time windows on a customer-based graph.

5.2.2 Time-Dependent VRP on customer-based multigraphs

Here, we consider an intermediate representation between $TDVRPs$ defined on customer-based graphs and those defined on road networks. We have a customer-based graph, but with multiple links between two customers that represent different paths in the road network obtained with different departure times. Thus, rather than having a single fixed path between two customers, alternative paths can be used, see Figure 5.2(a). In this figure, the nodes represent customers and the multiple links between two customers stand for alternative paths in the underlying road network, depending on the departure time.

We are only aware of heuristic methods in this case. The authors in [21] investigate a $TDVRP$ defined on a multigraph where parallel arcs stand for different time-dependent paths obtained with speed functions, thus satisfying the FIFO property. A tabu search is proposed to solve the problem with neighborhoods based on swapping two customers and reversing subsequences of customers. In [22], a tabu search is applied to a multigraph representation to address a $TDVRP$ with heterogeneous fleet. In this work, the insertion of a customer in a route must also integrate the selection of appropriate arcs between consecutive customers, which makes the problem more difficult. This is addressed with dynamic programming, but also with heuristic methods.

A multigraph representation is used in [95] for the $TDVRPTW$. When the sequence of customers in a route is determined, a Fixed Sequence Arc Selection Problem is solved using dynamic programming, to identify the arc (path) between each pair of consecutive customers in the route that must be selected (as it is done in [96] for a dial-a-ride application). In [23], the authors consider the time-dependent alternative vehicle routing problem with time windows. Two arcs are defined between each pair of nodes: the first arc is associated with a time-dependent travel speed distribution, to be used in case of low traffic, and the second one is associated with a constant travel time for heavy congestion hours. Thus, depending on the departure time, one of the two arcs dominates the other.

5.2.3 Time-Dependent VRP on Road Networks

In this section, the road network is fully exploited when considering time-dependent paths between two customers. That is, no abstraction takes place by first generating a customer-

based graph. Exact methods and heuristics for solving the $TDVRP_{RN}$ are reviewed in the following.

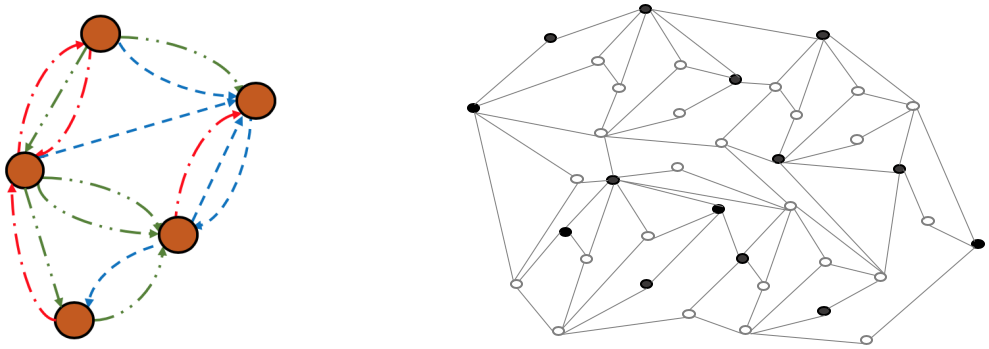
Heuristics

Using a Mixed-Integer Linear Programming (MILP) solver, the authors in [25] solve a $TDVRP$ with path flexibility under stochastic and deterministic conditions. The objective is to minimize the total expected cost. The authors compute, off-line, a non-exhaustive subset of shortest paths between each pair of customers by selecting different departure times and using a time-dependent Dijkstra's algorithm. Each path is associated with a decision variable in the MILP. This approach solves exactly an approximation of the problem because not all shortest paths are computed. Thus, it can be considered as a heuristic.

In [24], the authors address a time-dependent VRPTW on a real network made of two categories of arcs: main roads with time-dependent travel speeds, and small streets with constant speeds. Time-dependent travel speeds on arcs are created with high degree polynomial functions. A Greedy Randomized Adaptive Search Procedure is used to solve the problem. It enables savings of 12.5% in travel time when compared to solutions obtained with constant travel times.

Exact methods

To the best of our knowledge, there is no exact method for the $TDVRP_{RN}$ (without time windows). However, an exact method for the $TDVRPTW_{RN}$ is reported in [95]. The authors adapt a branch-and-price algorithm previously developed for a time-independent variant. The pricing problem is a Time-Dependent Shortest Path Problem with Resource Constraints, which is addressed with a time-dependent labeling algorithm based on a bi-directional search strategy that limits the number of generated labels. To account for the road network, where an arc can be traversed several times and by more than one vehicle, the branching scheme is defined as follows: 1) select an arc (i, j) with fractional flow ϕ_{ij} and 2) generate two branches by setting either the flow upper limit to $\lfloor \phi_{ij} \rfloor$ or the flow lower limit to $\lceil \phi_{ij} \rceil$. Unfortunately, it can happen that all flows are integer, while the routing solution is fractional. In this situation, an alternative binary branching scheme is applied. In the first branch, it is checked if the flow support graph of the current fractional solution (i.e., all arcs with a positive flow) contains a solution by enumerating all feasible routes and by solving a set covering formulation. In the second branch, at least one arc not in the flow support graph is enforced. The results of the branch-and-price algorithm on the road network graph are compared to those of two customer-based graphs where the single path between two customers is based



(a) Multigraph (b) Road network graph

Figure 5.2 Road Network representations

either on travel distance or travel time. The comparison shows that the solution cost on the road network provides average improvements of 1.7% and 7.3% over the min-distance graph and min-time graph, respectively.

5.3 Problem Description

In the following, we first define the time-dependent road network graph. Then, we introduce two problems defined on this graph, namely, the Time-Dependent Shortest Path Problem ($TDSPP$) and the $TDVRPTW_{RN}$.

5.3.1 Time-dependent road network

The road network is a directed graph $G = (V, E)$, where the set of vertices $V = \{0, 1, 2, ..., n\}$ corresponds to road junctions and the set of arcs E to road segments between pairs of junctions. Each arc (i, j) is associated with a distance d_{ij}, a time-dependent speed function $v_{ij} : t \rightarrow R^+$ that returns the speed at time t, and a time-dependent cost function $c_{ij} : t \rightarrow R^+$ that returns the cost of traversing arc (i, j) at time t (often, c_{ij} corresponds to the travel time). Each speed function is a stepwise function, from which a piecewise travel time function can be derived, see Figure 5.1.

5.3.2 $TDSPP$

A path p in the road network G from a source node $s \in V$ to a destination node $s' \in V$ is defined as a sequence of consecutive arcs $(i_1, i_2), (i_2, i_3), ..., (i_{j-1}, i_j)$ with $i_1 = s$ and $i_j = s'$.

Alternatively, the path can be viewed as the sequence of nodes $s = i_1, i_2, i_3, ..., i_{j-1}, i_j = s'$.

The $TDSPP$ then consists in identifying a minimum-cost path p from a source node $i_1 = s$ to a destination node $i_j = s'$, given a departure time t_0. The cost of path p at time t_0, $cp_p(t_0)$, is defined recursively as follows:

$$cp_{i_1,i_2}(t_0) = c_{i_1,i_2}(t_0), \tag{5.1}$$

$$cp_{i_1,...,i_j}(t_0) = cp_{i_1,i_2,...i_{j-1}}(t_0) + c_{i_{j-1},i_j}(t_0 + tp_{i_1,...,i_{j-1}}(t_0)), \tag{5.2}$$

where

$$tp_{i_1,i_2}(t_0) = tt_{i_1,i_2}(t_0), \tag{5.3}$$

$$tp_{i_1,...,i_j}(t_0) = tp_{i_1,i_2,...i_{j-1}}(t_0) + tt_{i_{j-1},i_j}(t_0 + tp_{i_1,...,i_{j-1}}(t_0)). \tag{5.4}$$

Note that $tp_p(t_0)$ is the travel time of path p at departure time t_0 and $tt_{ij}(t) = \frac{d_{ij}}{v_{ij}(t)}$ is the time-dependent travel time along arc (i, j) at time t.

5.3.3 $TDVRPTW_{RN}$

In the $TDVRPTW_{RN}$, a set of customers $C \subset V$ and a depot (node 0) are located on the time-dependent road network G, as defined above. Each customer $i \in C$ has a demand q_i, a time window for the service start time $tw_i = [a_i, b_i]$ and a service or dwell time s_i. A vehicle cannot arrive at customer i after the upper bound b_i of the time window, but can arrive before the lower bound a_i, in which case the vehicle waits until time a_i to start the service. The set of vehicles K, each of capacity Q, is located at the depot. The time window at the depot $[a_0, b_0]$ defines the beginning and end of the time horizon. The problem is then to generate a set of feasible vehicle routes (solution), one for each vehicle, that start and end at the depot and serve all customers at minimum cost. The latter is obtained by summing the route durations (travel time + waiting time + service time), where the travel time is time-dependent. A solution is feasible if it satisfies the capacity constraints and the time windows.

Note that the travel time of a path in the road network between two customers i and j for any given departure time t, $tp_{i,j}(t)$, is a piecewise linear function derived from the speed functions of all arcs along the path.

5.4 Time-dependent shortest paths

A time-dependent variant of Dijktra's algorithm is used to identify the minimum cost (travel time) path between a source customer s and a target customer s' in the road network G at a given departure time t_0, see Algorithm 1. In the pseudo-code, i denotes the current vertex; t_j is the arrival time at vertex j; $flag_j$ is *True* if vertex j is permanently labeled, that is, if the best path from customer s to vertex j has been found; and $label_j$ is the label of vertex j, that is, the travel time of the best path from s to vertex j. At the beginning, $label_s = 0$ and $flag_s = True$, while $label_j = \infty$ and $flag_j = False$ for $j \neq s$. At each iteration and starting from the current node, which is customer s at the beginning, the arrival time at each non-permanently labeled successor is calculated with the IGP procedure [14]. If the new path is better than the best known one, then the label is updated. In the special case where the successor is customer s', the new path is considered only if $t_{s'}$ does not exceed the time window's upper bound of s'. Once all successors of the current vertex have been considered, the vertex in the road network with the minimum label among all those that are not permanently labeled becomes the current one and its label is made permanent. This is repeated until customer s' is permanently labeled.

Algorithm 1 – Time-dependent Dijkstra's algorithm

Input: road network $G = (V, A)$, source s, target s', departure time t_0
Output: time-dependent path of minimum travel time from s to s'
$label_s = 0$
$flag_s = True$
for $j \in V$, $j \neq s$ **do**
    $label_j \leftarrow \infty; flag_j \leftarrow False$
$i \leftarrow s$
$t_i \leftarrow t_0$
while $flag_{s'} \neq True$ **do**
    for every successor j of i with $label_j = False$ **do**
        $t_j \leftarrow Algorithm\ 2(G, (i, j), t_i)$
        if $j = s'$ **then**
            if $t_j \leq b_j$ **then**
                $label_j \leftarrow min\{label_j, t_j\}$
        else
            $label_j \leftarrow min\{label_j, t_j\}$
    $i \leftarrow argmin\{label_j \mid j \in V \text{ and } flag_j = False\}$
    $flag_i \leftarrow True$

As previously mentioned, the path from s to the current node is extended to all non-permanently labeled successors at each iteration of the time-dependent Dijkstra's algorithm. The extension from the current node to a given successor through a particular exit arc is done with the IGP procedure, using the speed function associated with that arc, to produce the arrival time at the successor, see Algorithm 2. In the pseudo-code, *lb(period)* and *ub(period)* denote the lower and upper bounds of a time period and we assume that the associated speed is defined over the interval $[lb(period), up(period)[$ (i.e., the speed changes at the upper bound). After identifying the time period, and thus the speed, associated with the departure time t on arc (i, j), the procedures moves from one period to the next until the distance traveled d^+ exceeds d_{ij}. At that point, the arrival time at j is obtained by adding to the lower bound of the current period, which is stored in t, the time needed to travel the remaining distance to node j.

Algorithm 2 – Arrival time at a successor of a node

Input: road network $G = (V, A)$, arc (i, j), departure time t
Output: arrival time at j
$period \leftarrow 0$
while $(t \geq ub(period))$ **do** $period \leftarrow period + 1$ ▷ find time period of t
$d^- \leftarrow 0$
$d^+ \leftarrow (ub(period) - t) \times v_{ij}(t)$
while $d^+ \leq d_{ij}$ **do**
    $t \leftarrow ub(period)$
    $d^- \leftarrow d^+$
    $period \leftarrow period + 1$
    $d^+ \leftarrow d^- + (ub(period) - t) \times v_{ij}(t)$
$t \leftarrow t + (d_{ij} - d^-)/v_{ij}(t)$

5.5 Problem-solving method

Our solution approach is based on the tabu search metaheuristic. First introduced by Glover [97], tabu search is a neighborhood-based metaheuristic widely used for solving vehicle routing problems [98]. First, an initial solution is generated with a simple heuristic and becomes the current solution. Then, at each iteration, the best solution in the neighborhood of the current solution is selected (even if worse than the current solution) and becomes the new current solution. This is repeated until a stopping criterion is satisfied, at which point the best solution visited during the search is returned. To avoid cycling, it is forbidden or tabu to perform certain moves that could lead back to a previously visited solution. In our case,

the initial solution is produced with a greedy insertion heuristic and the neighborhood of the current solution is generated using CROSS exchanges [99], see Section 5.5.2. Since evaluating this neighborhood is computationally expensive, different techniques are used to assess the feasibility and approximate the value of neighborhood solutions in constant time. The main components of our tabu search will now be described in the following.

5.5.1 Initial solution

The initial solution is generated with a greedy insertion heuristic, where the routes are constructed sequentially. At each iteration, a customer is randomly selected and inserted at the best feasible insertion place in the current route. When a route cannot admit any new customer due to capacity or time window constraints, a new route is constructed. This is repeated until all customers are visited. The greedy heuristic exploits the techniques developed for assessing the feasibility of an insertion in constant time (see Section 5.6), although each insertion is evaluated exactly by propagating its impact along the route.

5.5.2 Neighborhood structure

The tabu search exploits a neighborhood structure based on CROSS exchanges [99]. This type of exchange is well suited for problems with time windows because it does not reverse segments of routes. The CROSS exchange consists in swapping sequences of customers of arbitrary length, up to a maximum length L, between two routes, where the number of customers in the two sequences does not need to be the same. Parameter L was set to 8, based on the work in [99]. A CROSS exchange is illustrated in Figure 5.3, where a sequence of n_1 customers in the first route is exchanged with a sequence of n_2 customers in the second route, by removing arcs (i_l,i_{l+1}), (j_h,j_{h+1}), $(i_{l+n_1},\ i_{l+n_1+1})$, $(j_{h+n_2},\ j_{h+n_2+1})$ and by adding arcs (i_l,j_{h+1}), (j_h,i_{l+1}), $(i_{l+n_1},\ j_{h+n_2+1})$, $(j_{h+n_2},\ i_{l+n_1+1})$. In the figure, the squares are copies of the depot that represent the start and end of each route.

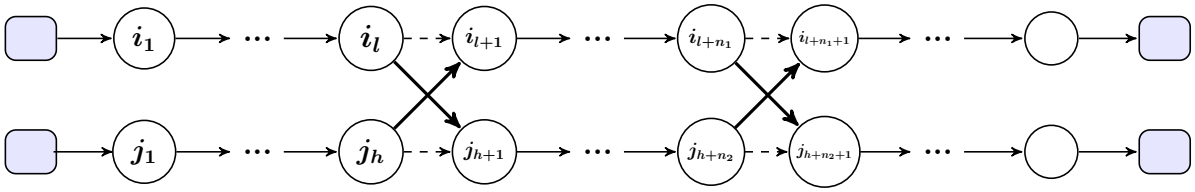


Figure 5.3 CROSS exchange

The neighborhood is explored in a systematic way by considering all possible exchanges of sequences of length $n_1 = 1, 2, ..., L$ and $n_2 = 1, 2, ..., L$ for every pair of routes in the current

solution. Due to the size of this neighborhood, the feasibility of a neighborhood solution as well as its approximate cost are evaluated in constant time, as discussed in Section 5.6.

5.5.3 Tabu list

When a move is performed to get from the current solution to the next one, the inverse move is declared tabu for a number of iterations, which is called the tabu tenure (tab). That is, if an exchange is performed at a given iteration ($iter$), then the exchange that leads back to the previous solution is tabu until iteration $iter + tab$.

5.5.4 Aspiration criterion

The classical aspiration criterion is used, where the tabu status of an exchange is overridden if it leads to a neighborhood solution that is better than the best known solution.

5.5.5 Diversification strategy

Diversification is used in tabu search to favor the exploration of new regions in the solution space when search stagnation is observed. The latter is detected when a number of consecutive iterations is performed without improving the best known solution. Diversification is realized by using an alternative objective, namely, minimization of the total distance, for a certain number of iterations. In such a case, the shortest paths in the road network between two customers are based on distance while the time-dependent travel times are only taken into account to guarantee feasibility (i.e., any path from customer i to customer j should reach j at or before its time window's upper bound b_j). Obviously, the solution value is also based on the total distance. After a certain number of iterations with the distance objective, the original objective (duration) is restored until stagnation occurs again. The parameter values for the number of iterations during diversification and number of consecutive iterations for detecting stagnation are discussed in the computational results, see Section 5.7.

5.6 Constant time evaluation framework

This section describes how the feasibility and approximate cost of a solution in the neighborhood of the current solution can be assessed in constant time. With regard to feasibility, we focus on time constraints, given that the vehicle load and, consequently, capacity constraints do not pose any difficulty.

It is important to note that additional information must be stored in the current solution

to achieve this result. The corresponding information needs to be updated (not in constant time) when the tabu search moves from the current solution to the next one. However, this is done much less frequently (once per iteration) than evaluating all solutions in the neighborhood of the current solution.

For the sake of simplicity, we assume in this section that service or dwell time $s_i = 0$ for every customer i.

5.6.1 Feasibility

The goal here is to avoid propagating the impact of a move along each modified route when a CROSS exchange is applied to the current solution, due to the size of this neighborhood. The additional information that needs to be maintained in the current solution for this purpose is derived from the so-called dominant shortest-path structure, as it is explained below.

Dominant shortest-path structure

Before running the tabu search, we generate different shortest paths for each pair i, j of customers by applying the time-dependent Dijkstra's algorithm with different departure times from i. A piecewise linear function is then calculated and associated with each path from i to j by combining the speed functions of all arcs that constitute the path. These piecewise linear functions return the arrival time at j for any departure time from i between a_i and b_i, see Figure 5.4. If we gather the functions of all paths between customers i and j, it is then possible to create a so-called dominant shortest path structure for the pair i, j by considering the best path at any moment in time. For example, $path2$ in Figure 5.4 is the best path between $t = a_i$ and $t = t_1$. Similarly, $path3$ is the best path between $t = t_1$ and $t = t_2$, etc. At the end, we obtain the structure shown in Figure 5.5, where the black dots at time $t = t_1$, t_2, t_3 correspond to crosspoints where the best path changes. This structure returns the arrival time at j for any given departure time from i, using the best path at the given departure time.

This dominant structure is used to evaluate the feasibility of a CROSS exchange, as it is explained below.

Bounds on departure times

Given the sequence of customers in a route of the current solution, the dominant shortest path structure associated with each pair of consecutive customers in this route can be used to determine the latest departure time at each customer to maintain feasibility from that

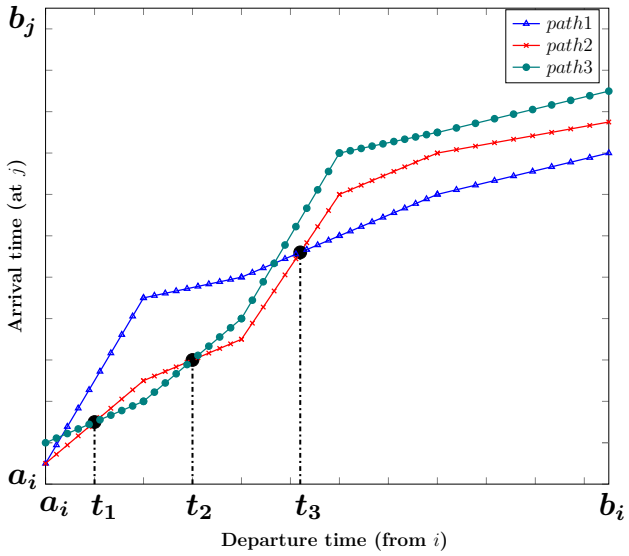


Figure 5.4 Piecewise linear arrival time functions for different paths between customers i and j

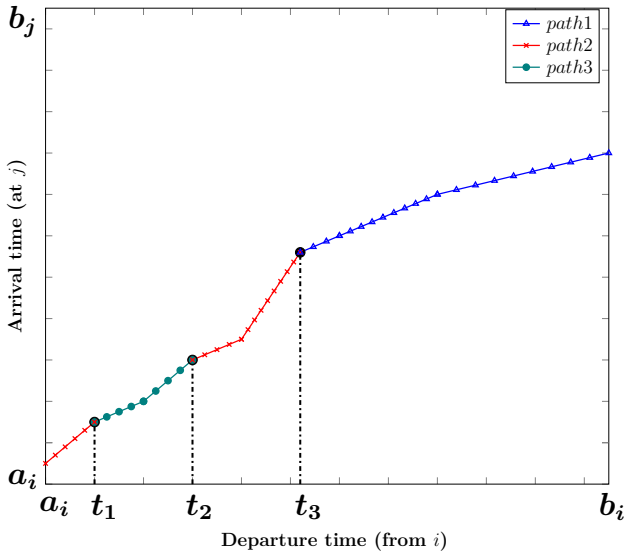


Figure 5.5 Dominant shortest path structure between customers i and j

customer up to the end of the route. Starting from the end depot and moving backward, the procedure is the following.

Let $i_0, i_1, i_2, ...i_{nr}, i_{nr+1}$ be a route r in the current solution with nr customers, where i_0 and i_{nr+1} are copies of the depot at the start and end of r. We also denote d_i the departure time from customer i in route r in the current solution. To compute the latest departure time ld_i from each customer i in route r we start with the end depot i_{nr+1} and end of time horizon

$ld_{i_{nr+1}}$. Based on the path p used in the current solution to go from customer i_{nr} to the end depot i_{nr+1}, we move backward along that path to infer $ld_{i_{nr}}$ from $ld_{i_{nr+1}}$. It should be noted that $ld_{i_{nr}}$ is reset to $b_{i_{nr}}$ when $ld_{i_{nr}} > b_{i_{nr}}$. Now, two cases can occur:

(a) If the dominant shortest path structure indicates that the best path from i_{nr} to i_{nr+1} at time $ld_{i_{nr}}$ is the same as the one used in the current solution at time d_i, then we continue moving backward, now with customer i_{nr} and time $ld_{i_{nr}}$.

(b) Otherwise, there is a better path than path p to go from i_{n_r} to i_{nr+1} at time $ld_{i_{nr}}$. In this case, we increase $ld_{i_{n_r}}$ as much as possible (i.e., without exceeding $b_{i_{nr}}$) to get as close as possible or match $ld_{i_{nr+1}}$ using the dominant shortest path structure of arc (i_{nr}, i_{nr+1}). In other words, the dominant shortest path structure allows us to identify the appropriate path and departure time at i_{nr} to reach the next node, here the end depot. Afterward, we continue moving backward with customer i_{nr} and the updated time $ld_{i_{nr}}$.

The above procedure is repeated until i_0 is reached, see Algorithm 3. From a practical point of view, the latest departure time is determined in case (b) by moving from one crosspoint to another along the dominant shortest path structure associated with customers i and j, until either b_i is reached, in which case b_i corresponds to the latest departure time at i, or the arrival time at j exceeds its latest departure time ld_j. In the second situation, ld_i is obtained by interpolating between the last and next-to-last crosspoints. This is illustrated in Figure 5.6, where ld_i^0 is the latest departure time at customer i obtained from ld_j by going backward through the path used in the current solution (it is assumed that this path is not $path3$, which is the best at departure time ld_i^0). We then successively move to ld_i^1 and ld_i^2 (where ld_j is exceeded) along the dominant shortest path structure. The latest departure time at customer i is finally identified by interpolating between ld_i^1 and ld_i^2.

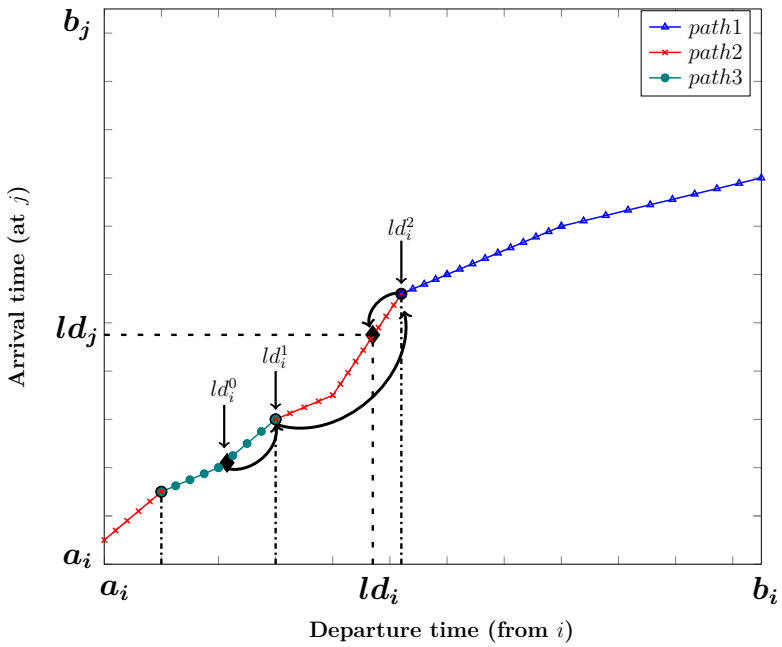


Figure 5.6 Latest departure time update

Algorithm 3 – Latest departure times

1: **Input**: road network $G = (V, A)$; route $i_0, i_1, ...i_{nr}, i_{nr+1}$, end of time horizon $ld_{i_{nr+1}}$ at end depot i_{nr+1}
2: **Output**: bounds on departure times at each node
3: **for** $k = nr, nr-1, ..., 0$ **do**
4: calculate ld_{i_k} from $ld_{i_{k+1}}$ by moving backward along the path p used to go from i_k to i_{k+1} in current solution
5: $ld_{i_k} = min\{ld_{i_k}, b_{i_k}\}$
6: **if** best path from i_k to i_{k+1} at time ld_{i_k} is not p **then**
7: find best path $p' \neq p$ and departure time ld_{i_k} to reach i_{k+1}, using dominant shortest path structure

---

CROSS exchange feasibility

Considering again Figure 5.3, where a neighborhood solution is obtained by exchanging two subsequences of customers between two routes in the current solution, the feasibility of this solution can be assessed in constant time. This is done as follows. First, the impact of replacing arc (i_l, i_{l+1}) by (i_l, j_{h+1}) in the first route is propagated from i_l to the customers in the sequence j_{h+1}, j_{h+2}, ..., j_{h+n_2} (which contains at most L customers) and then to i_{l+n_1+1}, using the dominant structure associated with each pair of consecutive customers. The arrival

time at i_{l+n_1+1} is then checked against its latest departure time to determine if the new route is feasible. Thus, there is no need to propagate until the end of the route. The same procedure is repeated for the second route involved in the exchange.

5.6.2 Approximate cost

To evaluate a neighborhood solution in constant time, we have no choice but to use an approximation. Essentially, for each customer i in a route of the current solution, we store a penalty p_i that stands for the delay incurred in the service start time of the next customer if the departure time at i is delayed by one time unit. Once the impact of a CROSS exchange, like the one in Figure 5.3, is propagated up to i_{l+n_1+1} (resp. j_{h+n_2+1}), the additional cost of the two new routes in the neighborhood solution is approximated by multiplying the delay $\Delta_{i_{l+n_1+1}}$ (resp. $\Delta_{j_{h+n_2+1}}$) by the penalty $p_{i_{l+n_1+1}}$ (resp. $p_{j_{h+n_2+1}}$). That is, the additional cost associated with this neighborhood solution is $\Delta_{i_{l+n_1+1}} \times p_{i_{l+n_1+1}} + \Delta_{j_{h+n_2+1}} \times p_{j_{h+n_2+1}}$. Note that Δ_i can well be positive or negative. For example, if a longer sequence is exchanged with a smaller one, then the delay can be positive on one route and negative on the other.

To alleviate the effect of the approximation, we keep the n_{approx} best solutions in the neighborhood, based on this approximate computation. Then, each one of these n_{approx} solutions is evaluated exactly, through propagation to the end of each route modified by the CROSS exchange, and the best solution obtained is selected as the new current solution. Through preliminary experiments, n_{approx} was set to 15 (values 5, 10, 15 and 20 were tested).

5.7 Computational experiments

In the following, we first describe the test instances used to perform our computational study. Then, a comparison is provided between our algorithm and the exact branch-and-price method BP reported in [95]. All experiments were performed on a Dell PowerEdge R630 server with two Intel Xeon processors E5-2637V4 with 4 cores and 128GB of memory each.

5.7.1 Test instances

The test instances, called $NEWLET$, come from a recent work in [95]. These instances were generated by the authors using a procedure for creating sparse graphs previously reported in [100]. Three graphs are available with $n = 50$, 100 and 200 nodes. Three different sets of random static travel times, called basic travel times, are associated with the set of arcs in each graph, based on different levels of correlation, namely, non-correlated (NC), weakly correlated

(WC) and strongly correlated (SC). This leads to $3 \times 3 = 9$ different networks. After dividing the time horizon into five periods, time dependency is accounted for by associating a speed profile with each arc in each network. This is done by multiplying the basic speed, which is derived from the basic travel time, by a multiplier in each period. It is possible to choose from three different sets of multipliers to define either congestion-free, normal or congested arcs. Then, 21 different categories of test instances are obtained by randomly selecting 16 or 33 customer nodes from the three 50-node networks ($3 \times 2 = 6$); 25, 33 or 50 customer nodes from the three 100-node networks ($3 \times 3 = 9$); and 25 or 50 customers from the three 200-node networks ($3 \times 2 = 6$). Finally, each category is duplicated by considering either narrow or wide time windows, for a total of 42 categories of instances. Since 5 different instances are available in each category, we have a total of 210 instances. The capacity of each vehicle is set to 200. The demands were produced as follows in [95]: first, routes were created with a greedy heuristic while considering only the time windows; then, the demand at each customer was generated to ensure that each one of those routes remained feasible. Finally, the service time at each customer is randomly selected in $\{1, 2\}$.

5.7.2 Comparison with optimal solutions

In the following, we first describe the procedure for tuning the main parameters of our algorithm. Then, we compare our solutions with the optimal ones reported in [95]. Finally, we analyze the impact of diversification.

It is important to note that the objective to be minimized in [95] is the total distance. Thus, at the road network level, minimum-distance paths are considered between two customers (or between the depot and a customer), while the time-dependent travel times are only used to guarantee path feasibility. In other words, minimum-distance feasible paths in the road network are looked for. At the route level, the set of vehicle routes must minimize the total distance. Consequently, we modified our algorithm by setting the total distance as the main objective and duration as the objective in the diversification phase.

Parameter tuning

Our proposed tabu search, called TS, has four main parameters, two of which are related to diversification: tabu tenure tab, maximum number of iterations it_{max}, number of consecutive iterations without improvement it_{cons} to initiate the diversification phase and number of iterations in the diversification phase it_{div}.

The following values were considered, where n_c denotes the number of customers:

- $tab = n_c/6,\ n_c/3,\ n_c/2,\ n_c$,
- $it_{max} = 5n_c,\ 10n_c,\ 15n_c$,
- $it_{cons} = n_c/2,\ n_c,\ 2n_c$,
- $it_{div} = n_c/10,\ n_c/5,\ n_c/2,\ n_c$.

Thus, we have a total of $4 \times 3 \times 3 \times 4 = 144$ possible combinations of parameter settings. An exhaustive evaluation of all those combinations was performed on a subset of 42 test instances. Precisely, one instance was randomly selected among the five instances available in each category. The parameter setting leading to the best average objective value on this subset was: $tab = n_c/3$, $it_{max} = 10n_c$, $it_{cons} = n_c$ and $it_{div} = n_c/5$.

In Table 5.1, we show how the average objective values evolve for each parameter when fixing the three other parameters at their best value. Each entry corresponds to the average gap in percentage between the distance $Dist(TS)$ produced by our Tabu Search and the optimum distance $Dist(Opt)$, that is:

$$Gap_O = \frac{Dist(TS) - Dist(Opt)}{Dist(Opt)} \times 100. \tag{5.5}$$

Considering the results for parameter it_{max}, it is clear that a sufficient number of iterations is required for the Tabu Search to converge. After $5n_c$ iterations, the best known solution is still far from the optimum. On the other hand, a total of $15n_c$ iterations does not bring any significant improvement over $10n_c$ iterations. For parameter it_{cons}, the value n_c is a good compromise between a larger number of diversification phases with shorter runs in-between ($n_c/2$) and a smaller number of diversification phases with longer runs in-between ($2n_c$). Otherwise, the value $n_c/3$ for parameter tab is clearly the best one, as is the value $n_c/5$ for parameter it_{div}.

Results on $NEWLET$ instances

Tables 5.2 and 5.3 report the results of our tabu search TS on the $NEWLET$ test instances for narrow and wide time windows, respectively. In these tables, each result was obtained on a single instance of each category, as defined by the graph size ($\#nodes$, $\#arcs$), number of customers ($\#cust.$) and correlation type ($Corr.$). Thus, the results here are based on 42 instances only, not $5 \times 42 = 210$ instances, because we had to choose in each category the single instance used by the authors in [95] to report their results. The optimum was obtained on all these instances within a computation time limit of 7,200 seconds in [95], except four

Table 5.1 Sensitivity analysis results

Parameter	$5n_c$	$10n_c$	$15n_c$	
it_{max}	7.9240	0.9285	0.9282	
	$n_c/6$	$n_c/3$	$n_c/2$	n_c
tab	2.449	0.929	2.139	2.331
	$n_c/2$	n_c	$2n_c$	
it_{cons}	1.258	0.929	1.154	
	$n_c/10$	$n_c/5$	$n_c/2$	n_c
it_{div}	2.940	0.929	1.109	1.589

instances with wide time windows. Tables 5.2 and 5.3 show the CPU time in seconds of TS ($TS\ Time$), the CPU time in seconds of the exact branch-and-price algorithm in [95] ($BP\ Time$), the improvement ($Impr.$) in percentage of the final distance produced by TS ($Dist(TS)$) over the initial distance produced by the greedy insertion heuristic ($Dist(Init)$), as defined by

$$Impr = \frac{Dist(Init) - Dist(TS)}{Dist(Init)} \times 100, \tag{5.6}$$

and the gap in percentage between the final distance obtained by TS and the optimal distance (Gap_O). Due to the slower machine used in [95], the CPU times of TS reported in the tables should be multiplied by a scaling factor of approximately 1.6.

Table 5.2 Results on *NEWLET instances* - Narrow time windows

Instance	# nodes	# arcs	# cust.	Corr.	*TS* Time(s)	*BP* Time(s)	Impr. (%)	Gap_{O} (%)
NEWLET	50	134	16	NC	0.640	95.6	3.020	0.761
				WC	0.967	1.7	1.712	0.751
				SC	0.296	1.0	1.846	0.781
			33	NC	2.813	3.5	5.610	0.822
				WC	2.945	14.4	3.860	0.816
				SC	2.694	8.9	2.823	0.850
	100	286	25	NC	2.719	1.1	6.713	0.874
				WC	1.935	1.4	2.528	0.922
				SC	1.429	1.5	5.727	0.930
			33	NC	2.488	1.0	1.704	0.968
				WC	2.982	2.0	1.492	0.964
				SC	3.264	49.6	2.019	0.942
			50	NC	5.251	4.0	2.898	0.992
				WC	5.464	4.0	1.826	0.982
				SC	7.151	201.3	1.961	0.962
	200	580	25	NC	2.393	4.0	3.071	0.948
				WC	2.731	4.0	2.986	0.883
				SC	2.406	3.9	1.833	0.991
			50	NC	17.931	13.0	1.547	0.929
				WC	27.182	7089.2	1.848	0.967
				SC	18.274	18.7	6.918	0.908

Concerning the results in Tables 5.2 and 5.3, we can see that TS significantly improves the solutions of the greedy insertion heuristic. The average improvement is 3.04% and 5.28% on the instances with narrow and wide time windows, respectively. More importantly, the gap between the final solutions produced by TS and the optimum is systematically under 1% on all instances, even if our algorithm was not intended to minimize the distance. The average gap on the instances with wide time windows is 0.843%, which is slightly better than the average gap of 0.902% on the instances with narrow time windows. It should also be noted that TS runs faster than BP on 11 out of 21 instances with narrow time windows and 17 out of 21 instances with wide time windows (after multiplying $TS\ Time$ by the scaling factor of 1.6). The average computing times in seconds of TS and BP are 5.43 and 358.27 over the instances with narrow time windows, and 13.14 and 2454.46 over the instances with wide time windows, respectively. Thus, TS is much faster on average, even after multiplying its computing times by 1.6. This is not a surprise, because exact methods are very sensitive to

the instance size. Also, they can sometimes be erratic and exhibit large computing times even on instances of relatively small size.

Table 5.3 Results on *NEWLET instances* - Wide time windows

Instance	# nodes	# arcs	# cust.	Corr.	*TS* Time(s)	*BP* Time(s)	Impr. (%)	Gap_{O} (%)
NEWLET	50	134	16	NC	1.561	15.0	6.772	0.751
				WC	1.526	2.8	7.556	0.716
				SC	1.463	1.7	7.083	0.743
			33	NC	4.769	7125.2	6.952	0.652
				WC	4.871	7200	6.693	–
				SC	2.163	7200	6.606	–
	100	286	25	NC	4.737	28.2	3.712	0.693
				WC	2.130	2.4	3.792	0.857
				SC	3.916	26.2	5.820	0.862
			33	NC	7.491	2229.9	6.681	0.882
				WC	5.766	5.1	4.570	0.901
				SC	5.865	24.4	4.548	0.909
			50	NC	13.051	6.1	3.774	0.945
				WC	17.366	56.7	2.769	0.845
				SC	11.774	1532.5	5.702	0.905
	200	580	25	NC	9.079	7199.2	4.693	0.822
				WC	8.481	96.6	6.914	0.966
				SC	9.316	27.4	5.628	0.901
			50	NC	63.811	7200	4.338	–
				WC	53.245	4364.3	2.511	0.974
				SC	43.602	7200	3.853	–

Impact of diversification

Here, we examine the impact of the diversification phase on solution quality. Tables 5.4 and 5.5 show the average gap with optimal solutions (Gap_O) for TS with and without diversification on the $NEWLET$ instances. These results show that diversification helps to significantly cut the gap with respect to the optimum, sometimes by more or close to one half (see the results with 100 nodes, 286 arcs, and 25 customers in Table 5.4 or those with 50 nodes, 134 arcs and 33 customers in Table 5.5).

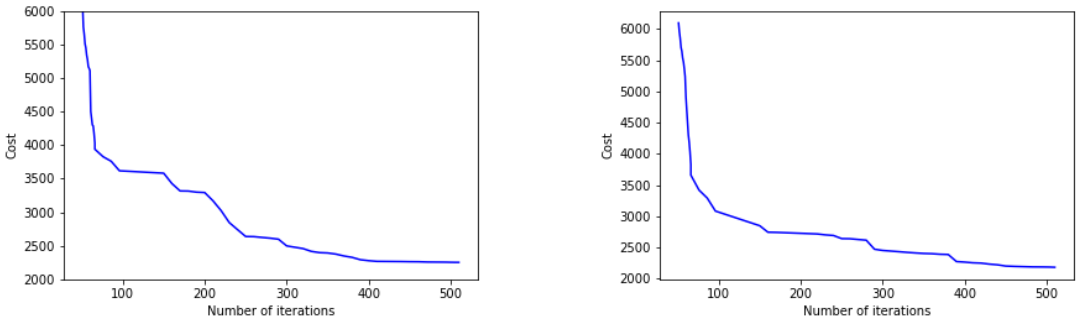
Table 5.4 TS optimal gaps with and without diversification - Narrow time windows

# nodes	# arcs	# cust.	Without diversification	With diversification
50	134	16	1.2560	0.7643
		33	1.3223	0.8293
100	286	25	2.0117	0.9088
		33	1.3730	0.9578
		50	1.1363	0.9784
200	580	25	1.2749	0.9406
		50	1.2009	0.9345

Table 5.5 TS optimal gaps with and without diversification - Wide time windows

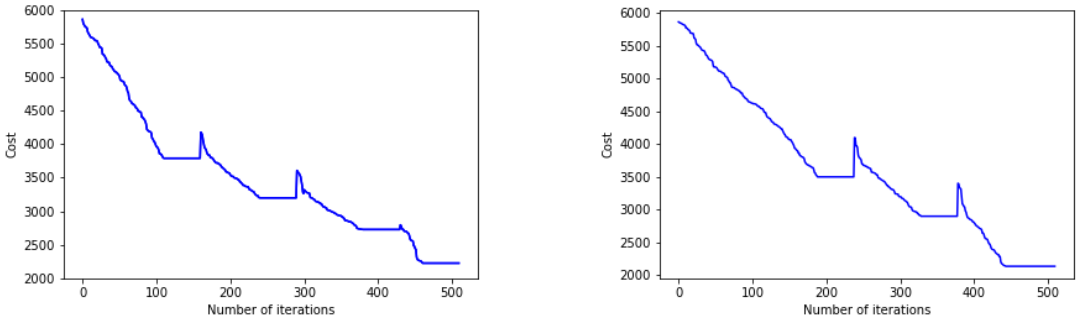
# nodes	# arcs	# cust.	Without diversification	With diversification
50	134	16	1.0202	0.7366
		33	1.1461	0.6520
100	286	25	1.1030	0.8040
		33	1.2300	0.8972
		50	1.4346	0.8983
200	580	25	1.1398	0.8963
		50	1.3710	0.9740

Figures 5.7 and 5.8 illustrate the evolution of the objective value (distance) on two instances with narrow and wide time windows, on a graph with 100 nodes, 286 arcs and 50 customers, while running TS without diversification and with diversification, respectively. Due to the range of the objective values, it is not possible to clearly see the final solution values and to claim that diversification is beneficial. However, this is indeed the case: on the instance with narrow time windows a total distance of 2246.8 is obtained without diversification, as compared to 2242.7 with diversification. Similarly, on the instance with wide time windows, a total distance of 2165.1 is obtained without diversification, as compared to 2154.6 with diversification.



(a) Narrow time windows (b) Wide time windows

Figure 5.7 Evolution of objective value of current solution without diversification



(a) Narrow time windows (b) Wide time windows

Figure 5.8 Evolution of objective value of current solution with diversification

5.7.3 Minimum duration objective

Our algorithm was primarily designed to minimize the total duration of the routes, which seems appropriate in the case of time-dependent travel times. Even if there is no alternative algorithm to compare with, we ran our Tabu Search on the 210 $NEWLET$ instances with this new objective (while minimizing the distance in the diversification phase). The results are reported in Tables 5.6 and 5.7, using the format of Tables 5.2 and 5.3 in Section 5.7.2. The two last columns (Gap_T and GAP_D) are calculated as

$$Gap_T = \frac{Time(TS_D) - Time(TS_T)}{Time(TS_T)} \times 100, \tag{5.7}$$

$$Gap_D = \frac{Dist(TS_T) - Dist(TS_D)}{Dist(TS_D)} \times 100, \tag{5.8}$$

to compare the solutions produced by TS when the distance (TS_D) or the duration (TS_T) is minimized. That is, Gap_T is the gap between the duration of the routes when the distance is minimized ($Time(TS_D)$) and when the duration is minimized ($Time(TS_T)$), while Gap_D is the gap between the total distance of the routes when the duration is minimized ($Dist(TS_T)$) and when the distance is minimized ($Time(TS_D)$). These two gaps indicate that the solutions obtained are significantly different depending if duration or distance is minimized.

Table 5.6 Results for *NEWLET instances* - NTW

Instance	# nodes	# arcs	# cust.	Corr.	TS Time(s)	Impr. (%)	Gap_T (%)	Gap_D (%)
NEWLET	50	134	16	NC	0.753	4.993	0.901	1.797
				WC	0.719	6.697	1.182	2.998
				SC	0.442	5.073	1.412	1.262
			33	NC	1.268	5.852	0.686	1.298
				WC	2.217	3.830	1.156	1.321
				SC	2.222	3.907	1.837	1.200
	100	286	25	NC	1.471	5.834	1.375	1.540
				WC	1.480	1.941	1.856	1.737
				SC	1.593	1.868	3.469	1.472
			33	NC	0.493	4.894	1.711	1.251
				WC	0.560	6.122	1.982	1.862
				SC	2.385	3.902	1.005	1.854
			50	NC	1.464	3.730	1.335	1.598
				WC	1.065	6.994	1.600	1.115
				SC	7.956	0.011	1.374	1.071
	200	580	25	NC	3.109	6.886	1.746	1.261
				WC	2.572	7.087	1.847	1.401
				SC	2.238	6.875	1.693	1.453
			50	NC	7.830	7.702	1.374	1.480
				WC	7.413	4.609	1.544	1.323
				SC	9.327	4.783	1.866	1.284

Table 5.7 Results for *NEWLET instances* - WTW

Instance	# nodes	# arcs	# cust.	Corr.	TS Time(s)	Impr. (%)	Gap_{T} (%)	Gap_{D} (%)
NEWLET	50	134	16	NC	0.778	7.985	0.843	2.994
				WC	0.730	7.711	1.334	4.370
				SC	0.734	2.619	1.118	3.446
			33	NC	2.189	6.869	1.468	1.233
				WC	2.271	5.838	1.855	2.450
				SC	2.227	4.874	1.764	1.086
	100	286	25	NC	1.499	5.088	1.577	1.620
				WC	1.009	7.916	1.690	1.905
				SC	1.681	5.852	1.710	1.327
			33	NC	2.594	5.001	1.130	1.496
				WC	2.517	5.025	1.520	1.208
				SC	2.315	7.805	1.134	1.118
			50	NC	3.507	5.814	1.011	1.268
				WC	7.723	7.149	1.814	1.810
				SC	8.334	6.955	1.779	1.052
	200	580	25	NC	2.464	8.137	1.071	1.896
				WC	2.431	8.931	1.122	1.614
				SC	2.438	5.084	1.453	1.326
			50	NC	8.403	5.031	1.960	2.467
				WC	8.220	7.916	1.702	2.791
				SC	9.812	6.010	1.240	2.368

5.8 Conclusion

This paper has proposed a tabu search heuristic, coupled with innovative techniques to evaluate neighborhood solutions in constant time, for the time-dependent vehicle routing problem with time windows on a road network. Computational experiments show that our method can identify high-quality solutions in very reasonable computation times on benchmark instances recently reported in the literature.

With regard to future work, we want to combine our tabu search with a previously developed deep learning neural network model that predicts travel speeds on the arcs of a road network, depending on a number of contextual variables. We will now apply this neural network in a real-time environment to continuously update the speed predictions based on current data. This update may, in turn, lead to a reoptimization of the planned routes using the tabu search.

Acknowledgments. Financial support was provided by the Natural Sciences and Engineering Research Council of Canada, while computing facilities were provided by Compute Canada. This support is gratefully acknowledged.

CHAPITRE 6 ARTICLE 3 : MANAGING IN REAL-TIME A VEHICLE ROUTING PLAN WITH TIME-DEPENDENT TRAVEL TIMES ON A ROAD NETWORK

Ce chapitre a été soumis pour publication sous forme d'article de revue scientifique : Gmira, M., Gendreau, M., Lodi, A., Potvin, J-Y. (2019). Managing in Real-Time a Vehicle Routing Plan with Time-Dependent Travel Times on a Road Network. *European Journal of Operational Research,* submitted for publication.

Abstract. Nowadays, devices like geographic information systems, global positioning systems, traffic flow sensors and cellular phones are a source of real-time traffic data like periodic estimates of travel times in a road network. However, many vehicle routing algorithms do not account for real-time changes in the road network. Accordingly, this paper considers the problem of adjusting in real-time a time-dependent delivery plan to respond to changes in travel times. This is achieved by either modifying the path used to go from one customer to the next in the road network or by modifying the sequence of customers in the planned routes. We also consider a variant of the problem in which some deliveries can be canceled. Computational results obtained by running simulations on a real road network for instances with up to 500 customers are reported and analyzed.

Keywords. Vehicle routing, road network, dynamic travel times, metaheuristics, tabu search.

6.1 Introduction

Vehicle routing has been a core logistic problem since introduced by Dantzig and Ramser [101]. Over the last 60 years, numerous papers have dealt with a large number of variants of this basic problem (see, for example, [102]). While early vehicle routing papers considered problems in which all inputs are static, it is now commonly accepted that, especially in urban areas, travel times do vary during a typical planning horizon, e.g., a day. This has led to a significant body of literature on time-dependent routing problems [93]. Recently, some authors have pointed out the fact that to be truthful to reality, time-dependent problems should not be defined with respect to the graph of customers, but rather with respect to the underlying road network [103, 104].

It is well known that unexpected events, such as accidents and non-recurrent traffic congestion may have an important impact on travel times observed on a road network. Because of that,

delivery plans (i.e., solutions of vehicle routing problems) may not be directly implementable in actual conditions. It therefore becomes necessary to adjust delivery plans to travel times that are observed or forecast using the latest information. The purpose of this paper is to develop an optimization methodology to adjust time-dependent delivery plans in an effective fashion. Such a problem falls into the realm of Dynamic Vehicle Routing Problems (DVRPs), i.e., problems in which inputs are received and updated concurrently with the execution of the planned routes [105].

This paper is organized as follows. In Section 6.2, we review the main classes of DVRPs. Section 6.3 then provides a detailed description of the problem at hand. Section 6.4 presents the solution approach proposed to tackle this problem. In Section 6.5, we describe the comprehensive computational study that was performed to validate our solution approach and report computational results based on a real road network in Montreal. Finally, we conclude and state future research directions in Section 6.6.

6.2 Literature review

As mentioned earlier, the problem that we consider falls under the general category of dynamic vehicle routing problems. For more than twenty years, several researchers have been interested in this class of problems, thanks in particular to (1) the increase in computing power due to better hardware, including parallel hardware, and (2) the advent of metaheuristics that allow high-quality solutions to be obtained in reasonable computation times.

Several surveys and book chapters have dealt with DVRPs. A detailed taxonomy of these problems has been presented in [106] based on different criteria: (1) problem type, (2) logistics context, (3) transportation mode, (4) objective function, (5) fleet size, (6) time constraints, (7) vehicle capacity constraints, (8) ability to reject customers, (9) source of dynamism, (10) source of stochasticity, and (11) solution method. In [107], the authors focus on DVRPs with deterministic and/or stochastic data, where dynamic data refer to new customer locations, customer demands and travel times. In [108], DVRPs are classified according to information quality and evolution. The authors also review various notions relative to the degree of dynamism. The latter is based on 1) the ratio between the number of dynamic requests and the total number of requests, 2) the normalized average of disclosure times, and 3) the reaction time, which is the difference between the disclosure time of a request and the end of its time window. Other interesting survey papers can be found in [109, 110].

In the following, we focus on DVRPs where the source of dynamism either comes from new customer requests or from perturbations to the travel times. It should be noted that,

according to [106], about 80% of the papers published on DVRPs involve dynamic customer requests, while about 10% involve dynamic travel times. Only a few papers address other sources of dynamism, like vehicle breakdowns, cargo damage, etc.

6.2.1 Dynamic customer requests

Several papers have been published in the literature about vehicle routing problems where customer requests occur dynamically over the day and must be handled in real-time. Therefore, rather than an exhaustive review, we provide here a brief overview of how these studies have evolved over time.

At the beginning, purely reactive approaches were proposed to address these problems. Essentially, a new static problem is defined and solved each time a new request occurs (this is called periodic reoptimization in [108]). A good example is the work in [111] in which the authors tackle a courier service application where small parcels are collected at different customer locations and brought back to a central office. The authors have adapted a tabu search with adaptive memory, previously developed for a static version of the problem, to account for the occurrence of new dynamic requests. To allow quick response times, a parallel implementation is proposed. In a subsequent work [112], the diversion of a vehicle from its current destination is considered whenever a new request occurs in the vicinity of the vehicle's current location.

One way to alleviate the myopic behavior of reactive approaches is to accumulate a certain number of dynamic requests before proceeding to reoptimization. This is the approach reported in [113]. In this work, the horizon is divided into time intervals of equal duration and all requests that occur during a given time interval are handled only at the end of that interval using an ant colony algorithm. Another similar approach is found in [114], where new requests are not handled immediately but only after some delay.

A new trend has then emerged, aimed at exploiting any available stochastic information about future customer requests. This information is either considered implicitly or explicitly. In the former case, we refer to the double horizon approach reported in [115], where a different objective is used to optimize the last part of a planned route. Through this objective, time slacks are introduced into the routes to accommodate potential future requests. In the latter case, we refer to the multiple plan approach [116] where different solutions are considered depending on different scenarios for the occurrence of new requests. Another approach to account for future requests is to devise waiting strategies for the vehicles to catch any potential request that can occur in their vicinity, see for example [115].

6.2.2 Dynamic travel times

The literature related to vehicle routing with dynamic travel times is relatively scarce, when compared to dynamic customer requests. In [27], the authors tackle a dynamic single load pickup and delivery problem. They present a routing system to dispatch a fleet of vehicles according to customer requests that occur at random over a planning horizon. The system exploits online information about travel times from a traffic management center, while also considering the occurrence of new customer requests. In this problem, the travel times are time-dependent and are modeled using a combination of piecewise and smoothing functions to avoid discontinuities [94]. Due to the computational load, only a subset of possible paths between customers are monitored when changes to travel times occur.

In [28], a genetic algorithm is proposed to address the same type of problem. A traffic simulator is used to model changes in travel times due to traffic incidents. The algorithm was applied to a small test network with 25 nodes and 80 arcs. The results indicate that the total travel time is improved on average by 3.7% over an approach where the travel times are forecast in advance.

In [29], the authors address a dynamic pick-up or delivery vehicle routing problem with time windows and time-dependent travel times. They consider multiple vehicles with different capacities, dynamic service requests, and dynamic variations in travel times. First, the problem is formulated as a mixed integer linear programming model and solved exactly for small instances with up to 10 customers. Larger instances are solved with a genetic algorithm. The results show that accounting for dynamic travel times is worthwhile, even when all requests are known in advance, in scenarios where the travel times fluctuate significantly over the planning horizon.

In [30], a tolerance for lateness at each customer is introduced. When the delay exceeds the tolerance, reactive changes are applied to the existing routes. This work was then extended in [31] by allowing diversion of vehicles from their planned destination, thus opening up additional opportunities to serve new requests. In [32], a further extension is considered, where it is assumed that the current position of each vehicle is known at all time, not only when a new request occurs or when a vehicle arrives at a customer location.

In [33], travel times are derived through a queuing model for traffic flows, where vehicles are “served” by road segments. The queuing model captures the relationship between traffic flow, density and speed, from which travel times can be obtained. Dynamic perturbations to the travel times are then produced by changing the inputs of the queuing model. This queuing approach is compared with alternative approaches and its benefits are evaluated.

In [117], a technician routing problem with stochastic service and travel times is introduced and solved using a two-stage stochastic programming method. The authors consider a specific variant of the field service routing problem in which two types of customers are served: mandatory, i.e., they have to be served within a specified time window, and optional, i.e., they may be served (or not) within the planning horizon.

6.3 Problem Description

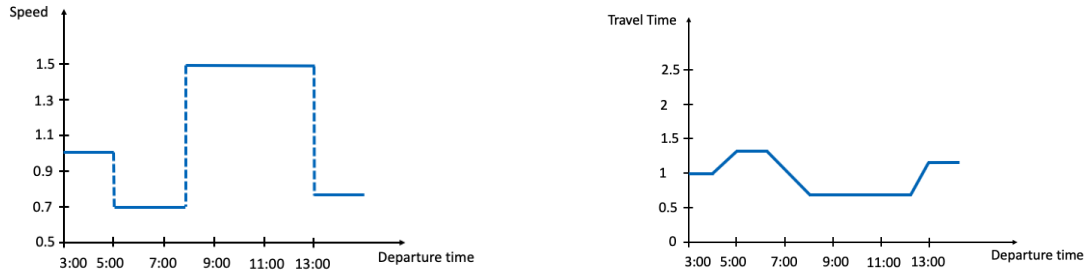
In this section, we first describe the planning problem that is used to generate the planned routes at the beginning of the day, before introducing its dynamic version in which the time-dependent delivery plan is adjusted in real-time to respond to changes in travel times. A variant of this problem is also considered in which some deliveries can be canceled.

6.3.1 Planning problem

We consider a vehicle routing problem with time windows and time-dependent travel times defined on a road network. In the following, we describe the road network and the time-dependent travel times, before formalizing the problem.

Road Network. The road network is a directed graph $G = (V, E)$, where the set of vertices $V = \{0, 1, 2, ..., n\}$ corresponds to road junctions and the set of arcs E to road segments between pairs of junctions. Each arc (i, j) is associated with a distance d_{ij}, a time-dependent speed function $v_{ij} : t \rightarrow R^+$ that returns the speed at time t, and a time-dependent cost function $c_{ij} : t \rightarrow R^+$ that returns the cost of traversing arc (i, j) at time t (often, c_{ij} corresponds to the travel time).

Travel Times. With each arc is associated a stepwise time-dependent travel speed function, where a different speed is associated with each time interval defined over the planning horizon. From this time-dependent travel speed function, a piecewise linear time-dependent travel time function can be derived, see Fig. 6.1.



(a) Speed function

(b) Travel time function

Figure 6.1 Piecewise linear travel time function derived from the stepwise speed function for an arc of length 1

A time-dependent variant of Dijkstra's algorithm can then use these arc-based functions to compute shortest paths (with regard to travel time) between any pair of nodes for any departure time [104].

Problem. Our problem involves a set of customers $C \subset V$ and a depot (node 0) located on the road network. Each customer $i \in C$ has a demand q_i, a time window $[a_i, b_i]$ for the service start time, where a_i and b_i are the lower and upper bounds of the time window, respectively, and a service or dwell time s_i. A set of vehicles K, each of capacity Q, is located at the depot. A vehicle cannot arrive at customer i after the upper bound b_i of the time window, but can arrive before the lower bound a_i, in which case the vehicle waits until time a_i to start the service. The time window at the depot $[a_0, b_0]$ defines the beginning and end of the planning horizon. The goal is to generate a set of feasible vehicle routes (solution), one for each vehicle, that start and end at the depot and serve all customers at minimum cost, where the cost corresponds to the total duration of the routes (i.e., travel time + waiting time + dwell time). A solution is feasible if it satisfies the capacity constraints and time windows. A solution is obtained by solving the planning problem with the tabu search heuristic reported in [104], where the upper bound of the time windows at each customer and the end of the time horizon cannot be exceeded (i.e., no lateness and no overtime are allowed). The number of vehicles is a decision variable in the planning problem.

6.3.2 Dynamic problem

The dynamic version of the problem is obtained by allowing dynamic perturbations to the time-dependent travel speeds. When a perturbation impacts one or more arc at a given time, the time-dependent travel speed function of each arc is modified, as indicated by the dotted

red lines in Fig. 6.2.

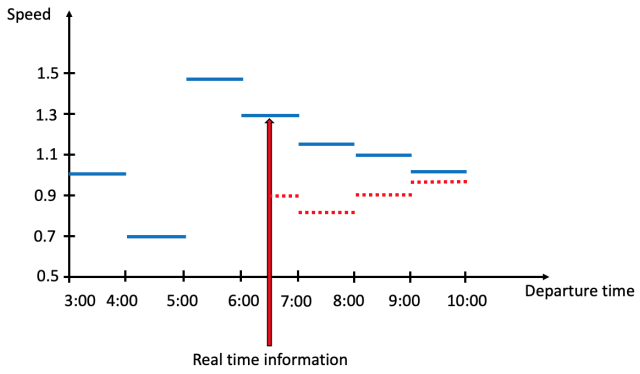


Figure 6.2 Example of a dynamic perturbation to a travel speed function

Clearly, if one of these arcs is part of the path leading from one customer to the next in a delivery route, the travel time between these two customers will change, with further impacts for the customers down the route, since the travel times are time-dependent. This is illustrated in Fig. 6.3, where a vehicle route starts from the depot, and while it is en route to its current destination (customer C_4), a travel speed update occurs, as indicated by the black dot. The system reacts by first recalculating the new arrival and departure times at customer C_4 from the current vehicle's position. This is repeated for each customer along the route, including the end depot. Eventually, it may be that the best path to reach the next location, starting with the current vehicle's position, will need to be recalculated, as it is illustrated in the figure. It may even be that the sequence of customers along the route has to be modified to account for the perturbation, as it is explained in Section 6.4.

In the dynamic version, it is not possible to guarantee anymore that the service will take place within the customers' time windows or that a vehicle will return to the depot before the end of the time horizon. Accordingly, the objective function of the dynamic problem minimizes the sum of travel time and a lateness penalty. The penalty is linear and is obtained by multiplying the total lateness by a penalty factor.

A variant of the dynamic problem is also considered where it is possible to cancel a customer if the lateness at this customer exceeds a given threshold. In this case, a large fixed penalty for each cancellation is added to the objective.

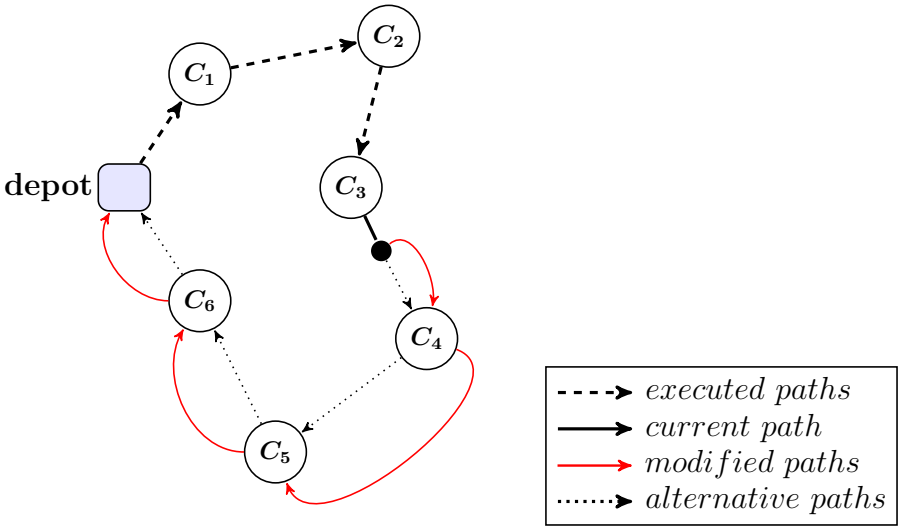


Figure 6.3 Dispatching of one vehicle in a dynamic setting (at the time of a new travel time update)

6.4 Dispatching system

The operations of the real-time dispatching system are based on the following assumptions:

- Real-time changes involve only travel speeds;
- Travel speed updates can occur at any time and involve a subset of arcs in the road network;
- The dispatching system can communicate at any time with a vehicle;
- The dispatching system knows the current location of each vehicle;
- When a customer is visited, this customer is removed from the planned route;
- Each time a travel speed update occurs, the current planned routes are reconsidered (which may imply reoptimization);
- During reoptimization, the exchange of customers between two routes is not allowed, since it is a delivery problem;
- During reoptimization, the current destination (customer) of each vehicle cannot change. Thus, no diversion is allowed.

Fig. 6.4 illustrates the main components of the dispatching system. We assume that we are working in an environment in which a first delivery plan was obtained by solving the planning problem described in section 6.3.1. As time goes on, new information on current and forecast travel times becomes available and needs to acted upon.

While the number of vehicles is a decision variable in the planning problem, it becomes fixed and cannot change in the dynamic setting. From the solution of the planning problem, we can compute the latest departure time at each customer in each route, which is a bound that guarantees feasibility of an entire planned route if the departure time from a customer in that route does not exceed its corresponding bound. These bounds are updated each time the solution is modified.

Once a speed update has been received, the planning problem that corresponds to the current status of the planned routes is addressed. Each vehicle route is considered in turn and new arrival and departure times are calculated at each customer. If the route is still feasible, then we keep it as it is. Otherwise, we recompute the best path between two consecutive locations in the route, starting with the vehicle's current location and its current destination and ending with the last customer and the end depot. As previously mentioned, this is done with a time-dependent variant of Dijkstra's algorithm.

Now, if the route is still infeasible, we try to improve it by modifying the sequence of customers (with the exception of the vehicle's current destination which is fixed) by applying a local search descent based on the Or-opt neighborhood [118]. That is, sequences of three, two and one consecutive customers are removed from the route and reinserted at the best possible place. This neighborhood is explored in a systematic way by first considering all possible sequences of three consecutive customers, two consecutive customers and a single customer, in this order.

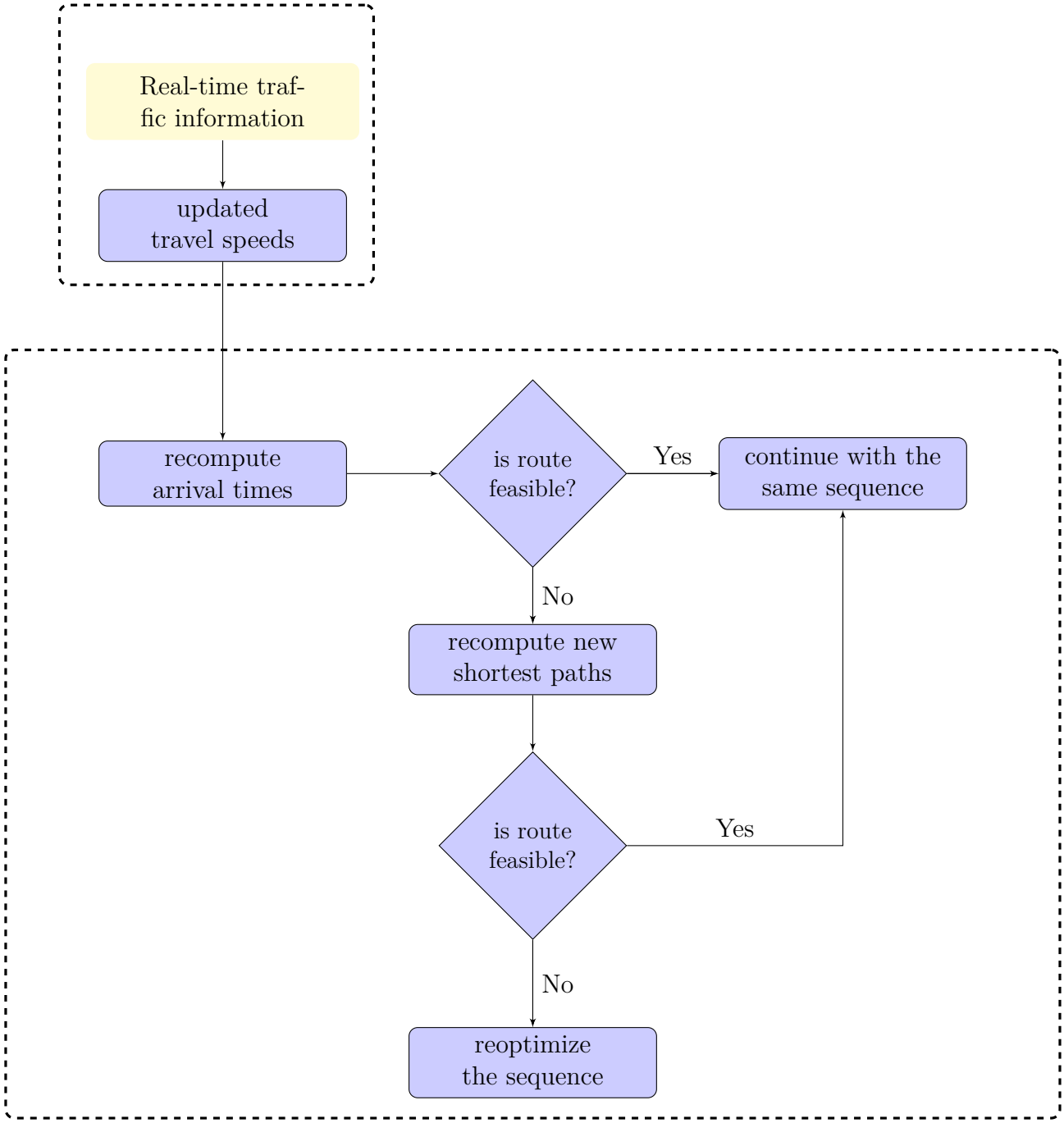


Figure 6.4 Real-time dispatching system

An example of an Or-opt with three customers is illustrated in Figure 6.5. In this example, the sequence of customers i_{l+1}, i_{l+2}, i_{l+3} is removed from the route and reinserted at another place, here between customers i_k and i_{k+1}. One advantage of Or-opt exchanges is that they do not reverse any part of the route, which is a desirable characteristic for problems with time windows. Clearly, once the sequence of customers is modified, the best path between two consecutive customers needs to be recalculated, starting with customer i_l in the figure. The final route obtained, which is a local minimum with regard to the Or-opt neighborhood, is then kept (feasible or not).

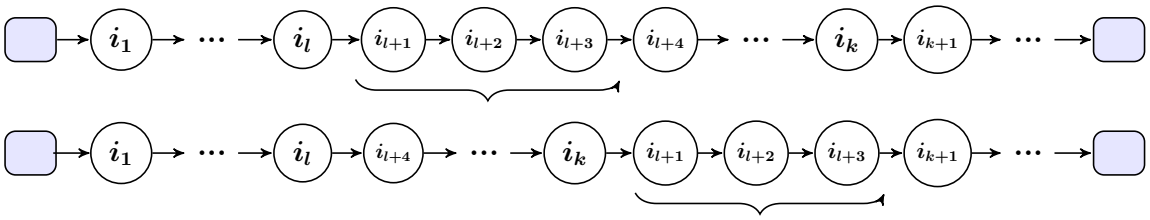


Figure 6.5 An Or-opt exchange of three customers

To summarize, we can distinguish three different cases when a reaction to a speed update occurs:

Case 1: the planned route remains feasible and the shortest paths and sequence of customers remain the same.

Case 2: the planned route is infeasible and feasibility is restored with changes to the shortest paths.

Case 3: the planned route is infeasible and feasibility is not restored with changes to the shortest paths.

Case 3.1: reoptimization with Or-opt restores feasibility of the planned route.

Case 3.2: reoptimization with Or-opt does not restore feasibility of the planned route.

6.5 Computational Study

In the following, we describe the comprehensive computational study that is used to validate our solution approach. We first describe the simulator developed to generate travel speed updates. Then, we introduce the test instances, which are based on a real road network. Finally, we report results obtained with our methodology with an increasing number of

customers. The code is written in Java and the experiments were performed on a Dell PowerEdge R630 server with two Intel Xeon processors E5-2637V4 with 4 cores and 128GB of memory each.

6.5.1 Simulator

We have developed a discrete-event simulator to generate travel speed updates and to maintain the status of the current solution (current location of each vehicle, current planned routes). A reaction takes place when speed updates occur. Their occurrence is modeled through an exponential distribution with an average inter-arrival time of 10 minutes.

The time-dependent travel speed function associated with each arc in the road network is a stepwise function, where a different speed is defined for each one-hour time interval over a planning horizon that extends from 9:00 AM to 7:00 PM. These travel speed functions are used to produce the initial routes at the beginning of the day and will be referred to as basic speed functions.

To generate travel time perturbations, we first randomly select one of the 12 districts that constitute the three central boroughs of Montreal used for this study (see Section 6.5.2). Then, the travel speed function of each arc (road segment) in the selected district is modified by applying a congestion rate between 1 and 5 to the basic speed in the corresponding time interval. The new speed is obtained by dividing the basic speed by the congestion rate. Thus, a lower rate stands for a minor incident while a higher rate stands for a major one.

It is assumed that an incident that occurs during a peak hour will have a major impact. Accordingly, the selection of the congestion rate of each arc is done probabilistically, using a probability distribution defined over the intervals [1, 2[, [2, 3[, [3, 4[and [4, 5], with a bias towards interval [4, 5]. Conversely, if an incident occurs during an off-peak hour, the probability distribution is biased towards the interval [1, 2[. The time span of an incident is also determined by the congestion rate. For example, if the latter is in the interval [4, 5], the basic speeds of the next three one-hour time intervals are also modified. If the congestion rate is smaller, then the number of affected time intervals is also smaller.

6.5.2 Test instances

The travel speed data used to generate the basic speed function of each arc in the road network were provided by a software development company that produces vehicle routing algorithms to plan the home delivery of large items (appliances, furnitures) to customers. Based on historical data that span approximately two years of operations, a previously developed

neural network model [119] was used to mine this information and produce basic travel speed functions for a typical operations day. These functions were then used to produce the initial planned routes at the beginning of the day.

The computational study is based on three central boroughs of Montreal with an important commercial activity and that are particularly sensitive to congestion, namely, Plateau Mont-Royal, Ville-Marie and Outremont, as shown in Fig. 6.6. These three boroughs contain over 4,400 road segments (arcs) and 2,150 road junctions (nodes). Plateau Mont-Royal has 6 districts, while Ville Marie and Outremont have 3 districts each, for a total of 12 districts.

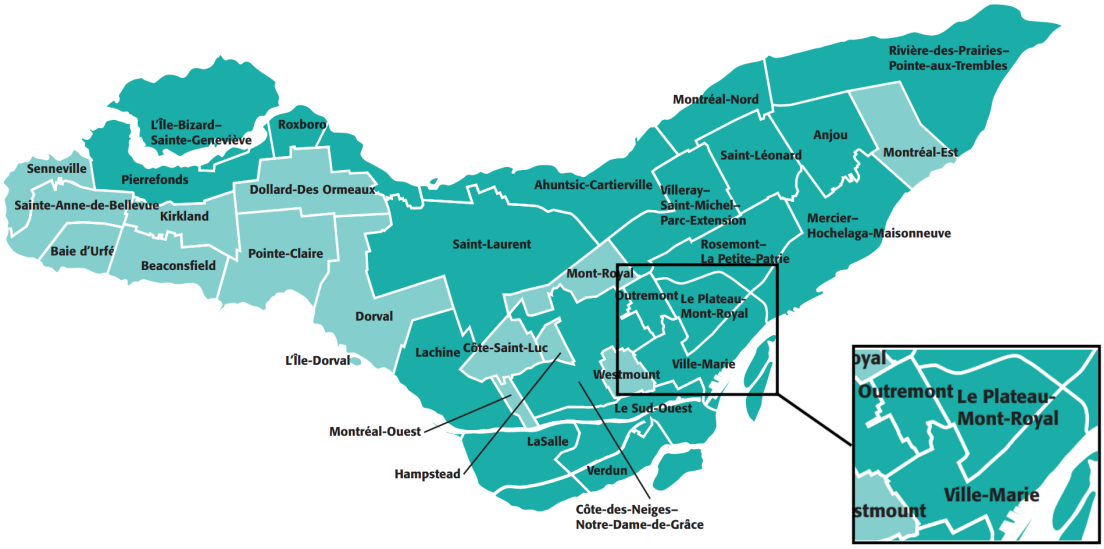


Figure 6.6 The three central boroughs used for the study

A central depot and a number of customers varying between 100 and 500 were randomly chosen in the above road network. More precisely, we have created five instances each with 100, 200, 300, 400 and 500 customers, for a total of 25 instances. For each customer i, the lower bound of the time window a_i was randomly chosen inside the planning horizon (while allowing sufficient time to return the depot from the customer). Narrow and wide time windows were produced according to [120], by setting the upper bound b_i in minutes to $(a_i + 3 \leq b_i \leq a_i + 4)$ and $(a_i + 10 \leq b_i \leq a_i + 15)$, respectively, while always taking care not to exceed the end of the planning horizon. The capacity of each vehicle was set to the number of customers in the instance multiplied by 8 (which means 800 for instances of size 100 and 4,000 for instances of size 500). The customer demands were generated between a lower bound of 20 and an upper bound of 70, by following the procedure reported in [103].

Finally, and without loss of generality, there is no service or dwell time at each customer.

6.5.3 Results

In the following, we report computational results obtained with our reactive procedure. With regard to the objective function used in these experiments (total route duration plus total lateness at customers plus total overtime at the end depot plus, possibly, total cancellation penalty), it should be noted that the lateness at each customer and the overtime at the end depot were multiplied by a penalty factor equal to 5 and that the cancellation of each customer led to a penalty of 1,000.

For comparison purposes, we consider an alternative non-reactive procedure which is the following. We consider the planned routes produced at the beginning of the day and, while travel speed updates occur along the day, we keep following the same planned routes and keep using the same path in the road network to go from one customer to the next. Thus, we do not react to the dynamic speed updates and we evaluate the objective value of this solution at the end of the simulation.

The results are reported in tables 6.1 and 6.2 and represent averages over the five instances of each size. The columns are the following:

- #Cust.: number of customers;
- #Cust./Route: number of customers per route;
- Scen.: in scenario 1, only lateness at customers is allowed; in scenario 2, cancellations are also allowed when the lateness exceeds a given threshold (set to 30 minutes);
- Comp. Time(s): total computation time in seconds for reacting to all speed updates in the dynamic setting;
- $Impr_1$: improvement in percentage of the objective value of the solution produced by the reactive procedure over the one produced by the non-reactive procedure, when starting with the initial feasible static solution:

$$Impr_1 = \frac{Non\text{-}Reactive - Reactive}{Non\text{-}Reactive} \times 100, \tag{6.1}$$

- $Impr_2$: improvement in percentage of the objective value of the solution produced by the reactive procedure over the one produced by the non-reactive procedure, when starting with a static solution that allows lateness and overtime (i.e. the objective used

to generate the initial solution is total duration of the routes plus total lateness plus total overtime, where the last two components are multiplied by the penalty factor); this improvement is calculated as in equation (1);

- Total Dur.: total route duration in minutes over all routes;
- Total Lat.: sum of lateness in minutes over all customers (thus, excluding the end depot);
- Total Over.: sum of overtime at the end depot in minutes over all routes;
- # Canc.: number of customer cancellations;
- # Late Cust.: number of customers that are served late.

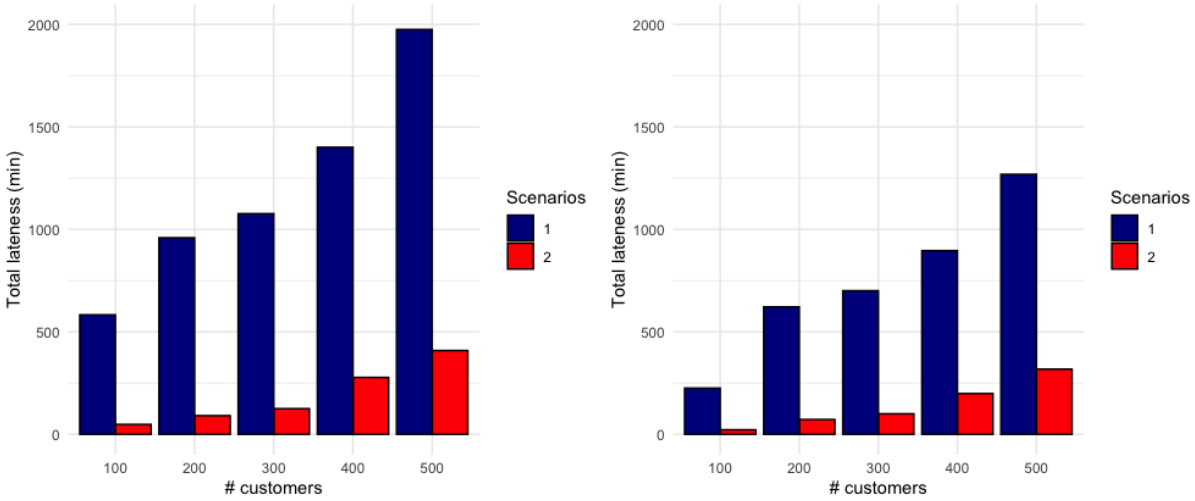
Table 6.1 Results for instances with narrow time windows

# Cust.	# Cust. /Route	Scen.	Comp. Time (s)	Total Dur.	Total Lat.	Total Over.	$Impr_1$ (%)	$Impr_2$ (%)	# Canc.	# Late Cust.
100	20	1	599.6	630.0	583.0	84.1	2.86	2.40	-	32
		2	421.5	582.2	48.3	0	9.64	-	8	2
200	38	1	817.3	1231.6	959.1	100.4	2.17	1.78	-	61
		2	406.4	1099.6	91.0	23.7	8.83	-	22	9
300	63	1	1427.6	1706.6	1076.6	132.5	2.25	1.91	-	85
		2	740.1	1411.4	125.2	53.4	8.34	-	28	15
400	79	1	2182.4	2227.8	1400.4	292.4	2.09	1.88	-	88
		2	917.7	1912.8	277.6	66.0	7.06	-	30	33
500	104	1	2985.9	2761.2	1975.4	365.1	2.46	2.07	-	102
		2	1695.8	2349.0	408.7	75.1	7.05	-	35	48

Table 6.2 Results for instances with wide time windows

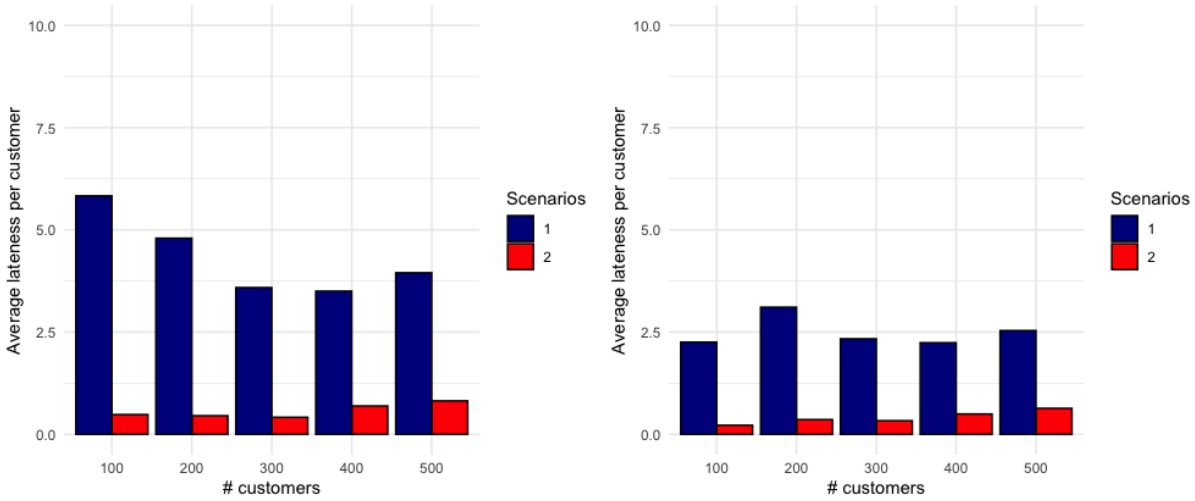
# Cust.	# Cust. /Route	Scen.	Comp. Time (s)	Total Dur.	Total Lat.	Total Over.	Impr_1 (%)	Impr_2 (%)	# Canc.	# Late Cust.
100	18	1	691.1	633.4	225.5	60.5	3.94	3.59	-	28
		2	383.8	585.3	22.1	0	11.83	-	8	2
200	36	1	944.8	1233.3	621.7	87.6	3.74	3.37	-	58
		2	591.4	1125.5	72.3	17.0	9.50	-	18	7
300	59	1	1622.3	1525.4	700.7	90.0	3.74	3.37	-	66
		2	1006.2	1420.4	100.0	39.1	8.29	-	21	9
400	74	1	2486.5	2027.0	896.1	209.2	3.10	2.76	-	89
		2	1140.0	1902.7	198.9	52.2	8.10	-	25	22
500	99	1	3222.3	2525.2	1268.6	246.0	2.95	2.66	-	92
		2	2187.1	2380.5	317.9	61.9	7.13	-	29	37

First, it should be noted that the number of customers per route increases with the total number of customers, in such a way that the number of vehicles stays around 5 in all cases. Clearly, the density of customers increases with the number of customers (since the service area remains the same), thus leading to smaller travel times between customers and thus, more customers in the planned routes. Not surprisingly, the total lateness and total overtime increase significantly from the smallest instances with 100 customers to the largest ones with 500 customers, particularly when the time windows are narrow since there is less flexibility for updating the shortest paths and optimizing the sequence of customers in the planned routes. The total lateness and total overtime also decrease significantly when customer cancellations are allowed (reduction factor of 7.5 and 6 for narrow and wide time windows, respectively). These trends are illustrated in Fig. 6.7 for the total lateness at customers. The average lateness per customer over all customers in the planned routes is also shown in Fig. 6.8. This average does not vary much with the instance size, but is clearly higher for instances with narrow time windows (overall averages of 4.3 and 2.5 minutes for narrow and wide time windows, respectively, for scenario 1). It also decreases sharply when cancellations are allowed since customers with high lateness can be removed from the planned routes.



(a) Narrow time windows (b) Wide time windows

Figure 6.7 Total lateness by number of customers and scenarios



(a) Narrow time windows (b) Wide time windows

Figure 6.8 Average lateness per customer by number of customers and scenarios

The improvement $Impr_1$ shows the benefits associated with the reactive procedure over the non-reactive one, when starting with the initial feasible static solution. The overall average improvement for instances with narrow and wide time windows is 5.3% and 6.2%, respectively. The improvement is larger for instances with wide time windows because there is more flexibility for reoptimization. When breaking these averages according to scenarios 1 and 2, we obtain 2.4% and 8.2% for narrow time windows and 3.5% and 9.0% for wide

time windows. Thus, the improvement is much larger when cancellations are allowed. This is easily explained since the planned routes produced by the reactive procedure do not contain all customers, due to cancellations, thus leading to shorter routes with smaller total lateness and total overtime. In a sense, the comparison between the two procedures is not fair in this case because all customers are visited with the non-reactive procedure. The improvement $Impr_2$ of the reactive procedure over the non-reactive one, when starting with the initial static solution that allows lateness and overtime, is smaller than $Impr_1$ (by about 15%), since better solutions are to be expected when the time windows are relaxed.

In the case of scenario 2, we can see in tables 6.1 and 6.2 that the percentage of canceled customers never exceeds 11% and 9% for narrow and wide time windows, respectively. However, as previously observed, it leads to large improvements with regard to total lateness and total overtime, when compared to scenario 1. Furthermore, the percentage of late customers decreases sharply from 25.3% in scenario 1 to 5.1% in scenario 2.

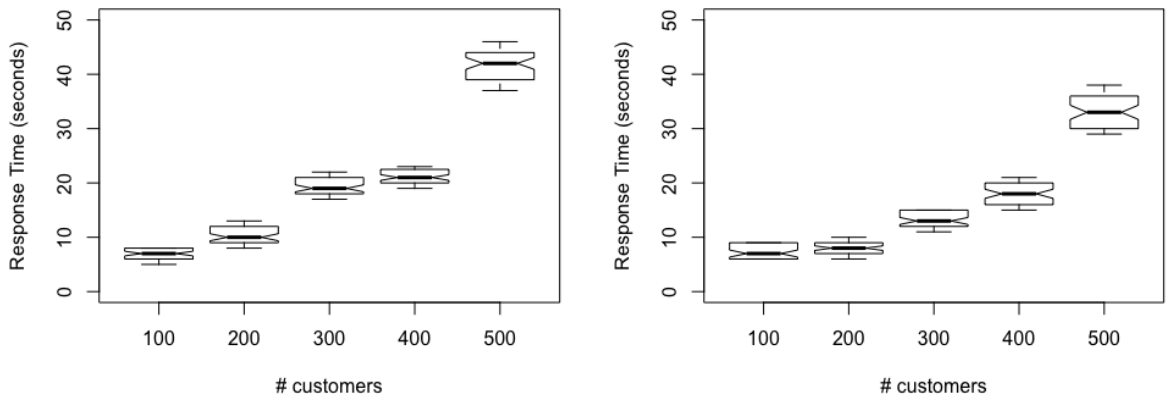
While Tables 6.1 and 6.2 report the total computation time for the reactive procedure over all speed updates, Tables 6.3 and 6.4 now show the average reaction time in seconds per speed update for narrow and wide time windows, respectively. Figure 6.9 provides an alternative view with boxplots that indicate the minimum, first quartile, median, third quartile and maximum values. As we can see, the maximum reaction time for the largest instances with 500 customers does not exceed one minute. Given that we did not put a full implementation effort to speed up the computations, the reactive procedure appears to be viable in a real-time context.

Table 6.3 Reaction time - Narrow time windows

# cust.	Min	Max	Mean
100	5	8	6.9
200	8	13	10.1
300	17	22	19.4
400	19	23	21.0
500	37	46	41.6

Table 6.4 Reaction time - Wide time windows

# cust.	Min	Max	Mean
100	6	9	7.4
200	6	10	8.0
300	11	15	13.2
400	15	21	18.0
500	29	38	32.9



(a) Narrow time windows

(b) Wide time windows

Figure 6.9 Reaction time

6.5.4 Impact of reoptimization with Or-opt

In this section, we evaluate the benefits of reoptimizing the planned routes with a local descent based on Or-opt exchanges. That is, we compare the original reactive procedure with a variant where the shortest paths can be recomputed but no reoptimization of the sequence of customers with Or-opt takes place. It corresponds to bypassing the box "reoptimize the sequence" in the illustration of the dispatching system in Fig. 6.4. The results are reported in Tables 6.5 and 6.6. In these tables, we show the results of the procedure without reoptimization. We also include the percentage of improvement $Impr_3$ of the procedure with reoptimization over the one without reoptimization, that is:

$$Impr_3 = \frac{Without\ Reoptimization - With\ Reoptimization}{Without\ Reoptimization} \times 100. \tag{6.2}$$

Overall, $Impr_3$ is equal on average to 0.74% and 1.01% for narrow and wide time windows, respectively. Although some improvements are obtained with reoptimization, they are relatively modest. We think that these results can be explained by the dynamic setting where frequent speed updates occur, thus quickly making a large part of the reoptimization work obsolete. On the other hand, if we focus on the total lateness and total overtime components of the objective, we observe improvements of 15.5% and 15.6%, respectively, over all test

instances. However, this is achieved by increasing the duration of the routes.

Table 6.5 Results with no reoptimization - Narrow time windows

# Cust.	Scenario	Comp. Time(s)	Total Dur.	Total Lat.	Total Over.	$Impr_3$ (%)	# Canc.	# Late Cust.
100	1	328.6	530.2	676.3	96.7	0.24	-	36
	2	205.3	552.9	62.8	18.5	0.43	8	4
200	1	322.0	1055.0	1130.8	117.5	0.50	-	70
	2	163.0	1084.9	111.9	27.7	0.78	22	11
300	1	456.8	1508.5	1268.2	157.7	0.54	-	98
	2	261.3	1390.7	153.0	62.4	0.84	31	37
400	1	816.2	1932.1	1677.7	345.0	0.75	-	101
	2	368.0	1865.3	338.72	75.9	1.19	32	38
500	1	979.4	2347.3	2372.4	433.4	0.88	-	119
	2	634.2	2222.0	560.0	86.7	1.27	37	54

Table 6.6 Results with no reoptimization - Wide time windows

# Cust.	Scenario	Comp. Time(s)	Total Dur.	Total Lat.	Total Over.	$Impr_3$ (%)	# Canc.	# Late Cust.
100	1	352.5	592.1	261.6	69.6	0.50	-	32
	2	186.5	575.5	25.7	13.3	1.03	8	3
200	1	453.5	1134.6	721.2	101.3	0.74	-	65
	2	277.9	1122.6	83.8	19.4	0.94	20	8
300	1	746.2	1418.6	812.8	104.8	0.75	-	75
	2	460.9	1416.1	116.0	44.6	1.17	23	11
400	1	1069.2	1872.9	1039.5	244.5	0.80	-	101
	2	501.6	1892.0	230.7	60.0	1.28	27	25
500	1	1256.7	2318.2	1471.6	287.8	0.93	-	105
	2	940.5	2374.9	368.8	71.6	1.97	31	43

6.5.5 Impact of cancellation threshold

We consider in this section the impact of the cancellation threshold, that is, the lateness value at a customer location beyond which cancellation can be considered, if it is beneficial with regard to the objective function. Tables 6.7 and 6.8 show the total lateness, total overtime,

and number of cancellations using three different threshold values of 15, 30 and 60 minutes for the instances with narrow and wide time windows, respectively. The trends observed in these tables are also illustrated in Fig. 6.10. Clearly, the total lateness, total overtime, and number of cancellations increase from the smallest instances with 100 customers to the largest instances with 500 customers. The total lateness and total overtime also increase from the smallest threshold value of 15 minutes to the largest value of 60 minutes, while the number of cancellations decreases. Furthermore, the differences between the solutions obtained with the three threshold values, either with regard to total lateness, total overtime or number of cancellations, tend to increase with the number of customers. The difference in the number of cancellations between a threshold of 15 minutes and a threshold of 30 minutes is particularly noticeable in this regard. Overall, these results indicate that the solutions obtained with our reactive procedure are quite sensitive to the value of this cancellation threshold.

Table 6.7 Impact of cancellation threshold - Narrow time windows

# cust.	Cancellation Threshold	Total Lateness	Total Overtime	# Canc.
100	15	31.50	0	12
	30	44.10	0	8
	60	62.79	7.72	4
200	15	80.91	16.61	33
	30	97.00	20.01	22
	60	123.38	38.49	12
300	15	98.67	44.13	51
	30	120.83	56.66	29
	60	148.84	76.03	20
400	15	116.06	57.93	69
	30	275.34	68.50	32
	60	293.22	83.62	30
500	15	367.11	66.00	87
	30	420.75	73.50	38
	60	483.18	96.17	34

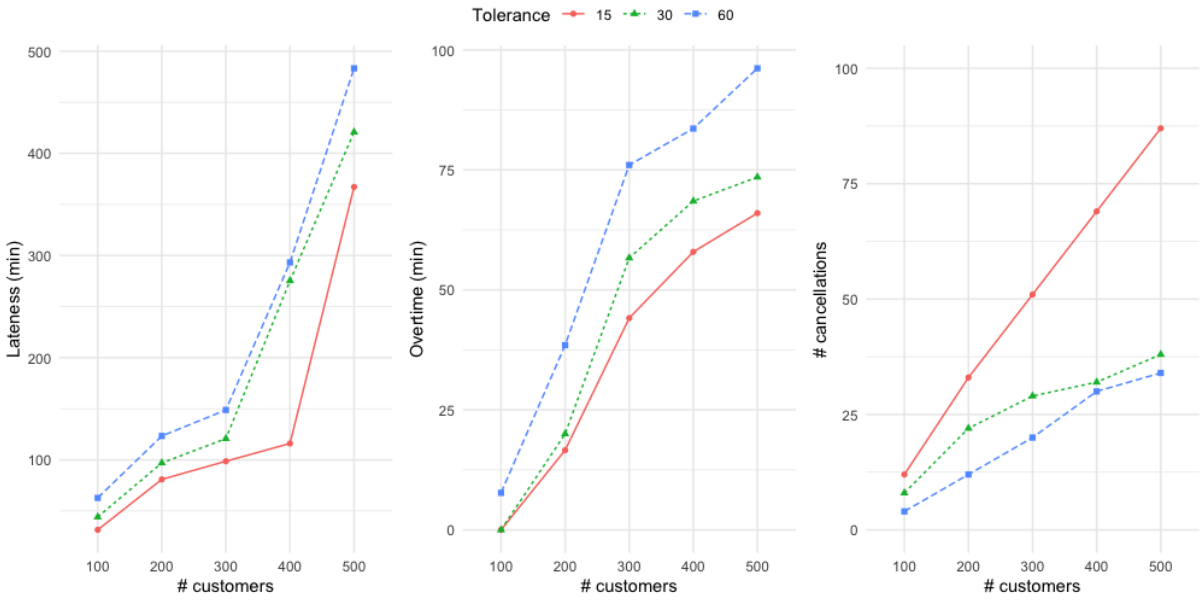
Table 6.8 Impact of cancellation threshold - Wide time windows

# cust.	Cancellation Threshold	Total Lateness	Total Overtime	# Canc.
100	15	18.10	0	11
	30	23.00	0	7
	60	39.02	5.99	3
200	15	65.38	12.36	31
	30	70.23	18.36	19
	60	101.35	30.07	10
300	15	83.19	29.73	46
	30	103.82	37.28	21
	60	121.26	58.92	14
400	15	152.04	41.10	58
	30	194.92	50.00	25
	60	219.11	71.16	18
500	15	267.21	49.96	76
	30	321.37	66.19	29
	60	400.72	81.20	25

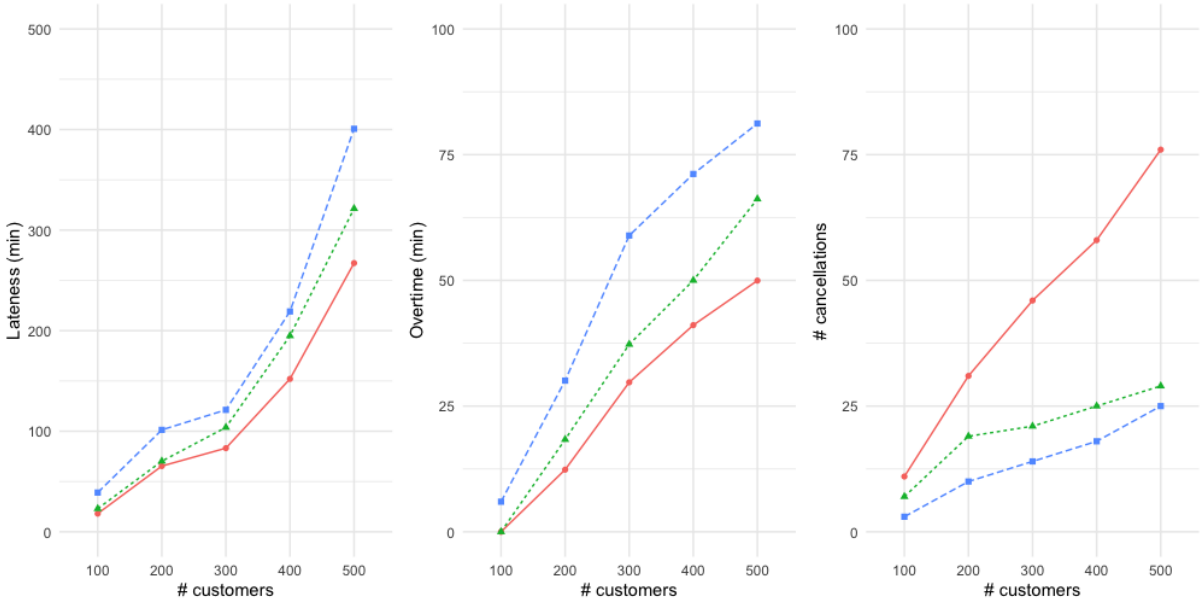
6.6 Conclusion

In this paper, we address a time-dependent vehicle routing problem with dynamic speed updates. A reactive procedure was devised that accounts for the time-dependent nature of the problem. The results obtained on a real road network show that improvements to the objective value are obtained when reacting to the travel speed updates. With regard to future work, we want to address a similar home delivery problem, but with a heterogeneous fleet. One can imagine larger vehicles that are used in the outskirt of a city, while smaller vehicles that are less sensitive to congestion are used downtown. Hence, different types of vehicles would be associated with different types of time-dependent travel speed functions. This problem can be compounded by allowing larger vehicles to transfer a part of their load to smaller vehicles, thus leading to synchronization issues.

Acknowledgments. Financial support was provided by the Natural Sciences and Engineering Research Council of Canada through its Canada Excellence Research Chair program. Computing facilities were provided by Compute Canada. This support is gratefully acknowledged.



(a) Narrow time windows



(b) Wide time windows

Figure 6.10 Impact of cancellation threshold

CHAPITRE 7 DISCUSSION GÉNÉRALE

Les trois articles exposés dans cette thèse constituent des étapes vers la planification et la gestion en temps réel de tournées de livraison. Ces articles y contribuent de façon différente : le premier évalue les paramètres fondamentaux des déplacements en milieu urbain menant à des profils de vitesse dépendants du temps. Ceux-ci sont fournis à un algorithme d'optimisation développé dans le second article qui permet de planifier, de façon quasi-optimale, des tournées de livraison pour la journée du lendemain. Ce plan sert également au pilotage en temps réel des livraisons à domicile. Ces trois articles sont indissociables par rapport à l'objectif ultime de la thèse, qui est la confection de tournées de livraison dans un réseau urbain.

En raison de l'application spécifique abordée dans cette thèse, soit les services de livraison à domicile en milieu urbain, il nous a fallu tenir compte de la taille considérable des réseaux routiers et des instances qui en découlent. Ces raisons nous ont poussés à développer des approches heuristiques et métaheuristiques de résolution afin de tirer profit des avantages qu'elles procurent au niveau des temps de calcul, mais également de la performance connue de certaines d'entre elles pour des problèmes de tournées de véhicules. D'ailleurs, le second article quantifie les performances de notre algorithme en comparant les solutions obtenues pour le problème de tournées de livraison avec des solutions optimales produites avec une méthode exacte [95]. Les écarts sont tous systématiquement inférieurs à 1%, avec des temps de calcul largement inférieurs à ceux de la méthode exacte. Ainsi, notre approche répond aux besoins de planification opérationnelle des services de livraison à domicile.

Le premier article avait montré qu'un modèle de réseaux de neurones récurrents de type LSTM permettait d'obtenir une précision de prédiction très intéressante, suggérant ainsi d'orienter davantage les recherches vers des algorithmes de prédiction basés sur de tels modèles afin de tenir compte d'informations historiques et de corrélations possibles entre les événements. Au regard de la méthodologie développée dans le troisième article afin de résoudre une version dynamique de notre problème de tournées de livraison, il serait certainement souhaitable d'intégrer un réseau de neurones capable de mettre à jour en temps réel ses prédictions de vitesses à partir d'informations sur le réseau routier également obtenues en temps réel. Actuellement, un simulateur est utilisé afin de générer les perturbations aux profils de vitesses.

L'outil présenté dans le chapitre 4 a pour objectif de prévoir, à court terme, les vitesses de parcours sur le réseau routier de la région métropolitaine de Montréal. Comme les méthodes d'IA sont désormais reconnues pour présenter certaines difficultés à prévoir les tendances,

puisque, par exemple, les vitesses diminuent globalement ou encore la congestion augmente avec le temps. Il serait donc opportun d'évaluer la qualité de prédiction de méthodes de prédictions à moyen et long termes.

De plus, deux algorithmes de partitionnement ont été utilisés pour générer des classes de segments routiers présentant des similarités du point de vue des profils de vitesses au cours d'une journée. Ces méthodes ont recours à des distances euclidiennes pour mesurer cette similarité. Cependant, les métriques euclidiennes présentent certaines lacunes pour le partitionnement de séries temporelles puisqu'elles ne considèrent pas la position des valeurs dans le temps (la séquence). D'un autre côté, des métriques telles que le *dynamic time warping* et le *cross-correlation distance* fournissent de bons résultats lorsqu'appliquées à des séries temporelles, cependant, elles présentent un inconvénient majeur au niveau des temps de calcul. Ainsi, les auteurs dans [121] rapportent des temps de calcul de la distance *cross-correlation* de l'ordre de 10 minutes pour 1000 vecteurs en entrée. Si une telle métrique était utilisée dans le chapitre 4, il faudrait sans doute plus de 1000 heures, ce qui n'est pas viable. Il serait donc intéressant de trouver des métriques de similarité alternatives, mieux adaptées aux séries temporelles que la distance euclidienne tout en relevant le défi d'obtenir des temps de calcul raisonnables dans le contexte de la prédiction.

Dans le chapitre 5, des échanges CROSS sont utilisés pour générer des solutions dans le voisinage. Nous évaluons systématiquement toutes les combinaisons possibles. Lorsqu'une recherche tabou est effectuée, nous observons que certains arcs sont souvent associés à une bonne solution et inversement d'autres arcs sont associés à de moins bonnes. Ainsi, sur la base d'un apprentissage au fil des itérations, il est possible de décider de ne pas enlever les arcs présentant un bon potentiel (au niveau du coût de la solution) et inversement, d'éliminer les arcs ayant un faible potentiel.

Dans le chapitre 6, un simulateur a été développé pour imiter les perturbations du réseau routier affectant les temps de parcours. Un taux de congestion variant entre les valeurs 1 et 5 est appliqué aux fonctions de vitesses produites par le réseau de neurones artificiels développé dans le chapitre 4. De nouvelles fonctions de vitesses sont obtenues en divisant les vitesses prédites par le taux de congestion. On obtient donc des vitesses très faibles lorsque le taux de congestion est élevé et inversement. Ainsi, ce simulateur fait l'hypothèse que toute perturbation ne peut qu'affecter négativement les temps de parcours, obtenus à partir de profils de vitesses prédits. Il serait intéressant de tenir compte de perturbations affectant positivement les temps de parcours, c'est à dire qui favorisent des vitesses plus élevées.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

8.1 Synthèse des travaux

Dans cette thèse, le premier aspect abordé, tel qu'exposé dans le chapitre 4, est celui de la prédiction de vitesses à partir de traces GPS recueillies par des capteurs embarqués dans des véhicules de livraison. Cette prédiction a été réalisée à l'aide d'une méthodologie basée sur un ensemble de techniques de forage de données (segmentation, imputation, etc.,) et des techniques d'apprentissages automatiques supervisé et non supervisé. L'approche que nous avons développée est en mesure de répondre efficacement à un contexte où les données recueillies sont volumineuses et nécessitent des moyens spéciaux pour les traiter.

Notre méthodologie a été testée sur un réseau routier réel de taille importante, soit le réseau de la région métropolitaine de Montréal, et a démontré une grande précision dans ses prédictions de vitesses. Les résultats obtenus dans cette première contribution confirment les conclusions de l'étude mentionnée en introduction [1] puisque l'erreur de prédiction en utilisant un réseau de neurones récurrent est nettement plus faible que celle produite par les méthodes statistiques utilisées à des fins de comparaison.

Le second aspect traité dans le chapitre 5 est la résolution par une approche métaheuristique (recherche tabou) d'un problème de tournées de livraison défini sur un réseau routier où les clients doivent être servis à l'intérieur de fenêtres de temps et où les vitesses sur les arcs du réseau varient selon le moment de la journée. Afin de tenir compte de l'aspect *time-dependent* du problème et de la structure du réseau routier, l'approche retenue intègre des techniques permettant d'évaluer en temps constant la réalisabilité des solutions dans le voisinage de la solution courante (en termes de respect des contraintes de fenêtres de temps et de capacité des véhicules), mais également de calculer en temps constant le coût approximatif de chacune de ces solutions de façon à ne retenir que les meilleures qui sont ensuite évaluées de manière exhaustive afin d'identifier la solution dans le voisinage qui deviendra la nouvelle solution courante. Combinées à une stratégie de diversification spécifique au problème traité, nous avons produit des solutions dont l'écart avec la solution exacte est systématiquement inférieur à 1%, pour des instances allant jusqu'à 580 arcs, 200 nœuds et 50 clients.

Enfin, le chapitre 6 traite d'une version dynamique du problème décrit dans le chapitre 5, où les vitesses sont mises à jour de façon dynamique, toutes les 10 minutes en moyenne. Cette nouvelle caractéristique engendre des défis importants tels que l'intégration de cette nouvelle information dans les routes courantes et des temps de calcul permettant une réaction en

temps réel. Les techniques développées au chapitre précédent qui permettent l'évaluation de la réalisabilité d'une solution en temps constant peuvent être également exploitées ici, bien qu'elles deviennent inutiles dès qu'une solution contient des retards, ce qui ne peut manquer de survenir suite à l'accumulation des mises à jour au fil du temps.

Différents types de réactions sont possibles suite à une mise à jour dynamique des vitesses : conserver la solution telle quelle, modifier les plus courts chemins entre les clients ou réoptimiser la séquence des clients visités dans les tournées courantes.

Le chapitre décrit également un simulateur d'événements discrets qui permet de générer des perturbations aux vitesses selon le moment de la journée et l'importance de l'incident. L'approche développée est testée sur trois arrondissements de l'île de Montréal où l'activité commerciale est importante. Les résultats obtenus sont très prometteurs puisqu'ils améliorent grandement une approche non réactive (aucune réaction aux mises à jour dynamiques, c'est-à-dire que les routes planifiées du début de la journée ne sont pas modifiées et sont parcourues telles quelles pendant toute la journée). Nous considérons également un autre contexte où l'annulation d'un client est considérée lorsque le retard à ce client excède un certain seuil. D'autres résultats intéressants concernent l'impact de ce seuil sur les solutions obtenues ainsi que l'amélioration apportée par la procédure de réoptimisation des tournées qui correspond à une méthode de descente avec voisinage Or-Opt.

8.2 Améliorations futures

Le fait de considérer un graphe qui représente directement la topologie d'un réseau routier (et non un graphe client ou un multigraphe) est en soi une idée assez récente et très peu de travaux ont été recensés jusqu'à présent sur ce sujet. En plus de notre approche, on rapporte seulement quelques heuristiques et une méthode exacte, ce qui laisse place à plusieurs avenues de recherche et développements possibles.

Plus spécifiquement, la solution proposée au chapitre 6 possède certaines limitations qui pourraient être corrigées en introduisant les développements suivants :

- permettre de détourner un véhicule de sa destination courante afin de servir un client qui apparaît non loin du véhicule (diversion);
- considérer une flotte hétérogène où les vitesses sur les arcs du réseau diffèrent selon le type de véhicules. On peut ainsi imaginer de plus gros véhicules pour desservir la périphérie d'une ville et des véhicules de plus petite taille, moins sensibles à la congestion, pour desservir les zones centrales. On peut même imaginer des points de

transfert permettant à un plus grand véhicule de transférer une partie de sa charge à un plus petit véhicule, entraînant de ce fait des contraintes de synchronisation entre eux;

- tenir compte des pauses réglementaires des conducteurs;
- intégrer d'autres aspects dynamiques touchant à la mise à jour des fenêtres de temps ou à l'occurrence de nouveaux clients, par exemple.

Un aspect important abordé dans le chapitre 6 touche à la contribution de la procédure de réoptimisation des tournées dans la qualité de la solution obtenue. Le constat est que la méthode de descente avec voisinage Or-opt permet certes d'améliorer la solution, mais pas plus de 0,744% et 1,011% respectivement pour les instances avec des fenêtres de temps étroites et larges. Ceci peut s'expliquer par la fréquence des mises à jour qui rend obsolète une grande partie du travail de réoptimisation. Il serait intéressant de tester des procédures de réoptimisation encore plus puissantes afin de supporter davantage cette hypothèse.

Nous avons eu recours à un simulateur au chapitre 6 afin de générer les modifications dynamiques des vitesses sur le réseau routier. Un développement intéressant serait d'intégrer plutôt une méthode de prédiction des vitesses en temps réel basée sur le type de réseau de neurones développé dans le chapitre 4. Cet outil de prédiction serait une composante d'un système de gestion de flottes de véhicules comme illustré dans la figure 8.1. Cet outil de prédiction serait capable de traiter des données structurées et non structurées, qu'elles soient historiques ou reçues en temps réel afin de prédire les vitesses de parcours sur le réseau routier.

L'objectif ultime serait de concevoir un système intégré avec différentes composantes qui interagissent entre elles. Comme illustré dans la figure 8.1, on retrouverait une base de données historiques contenant, par exemple, les profils de vitesses relevés sur les différents arcs du réseau routier et les conditions météorologiques. Ces éléments constituent généralement une bonne base de prédiction des vitesses. Afin de renforcer le pouvoir prédictif du système, les vitesses prédites sont mises à jour en fonction des données collectées en temps réel telles que les vitesses instantanées transmises par les capteurs ou encore les rapports d'incidents routiers.

Un tel outil de prédiction serait certainement un atout majeur pour la méthode d'optimisation des tournées, qui comprend l'optimisation des chemins entre les clients, mais également l'optimisation de l'ordre de visite des clients dans chaque tournée. Cette optimisation serait appelée à chaque fois que des mises à jour sont reçues. Finalement, suite à toute procédure d'optimisation (ou de ré-optimisation), des métriques sont collectées comme le nombre de

livraisons annulées ou en retard, le retard chez le client ou au dépôt final. Ces métriques sont analysées à des fins d'apprentissage et d'ajustement de l'algorithme. On comprend aisément que chaque composante du système de gestion des flottes de véhicules est indispensable puisqu'elle interagit avec une autre, participant ainsi à l'amélioration globale du système.

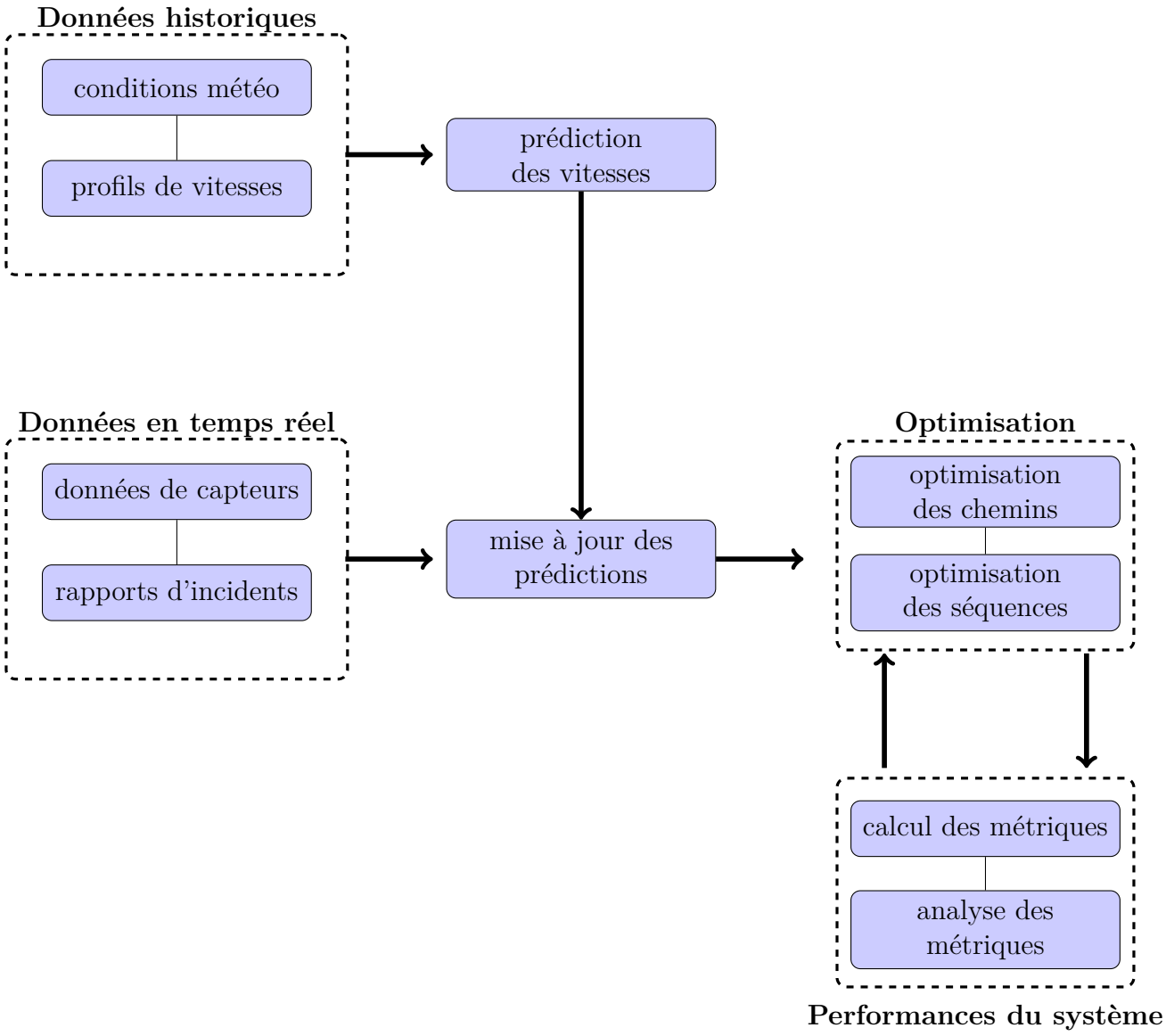


Figure 8.1 Système de gestion des flottes de véhicules intégré à une procédure de prédiction des paramètres de trafic

RÉFÉRENCES

[1] M. Chui, J. Manyika, M. Miremadi, N. Henke, R. Chung, P. Nel et S. Malhotra, "Notes from the ai frontier: Insights from hundreds of use cases," McKinsey Global Institute (Retrieved from McKinsey online database), 2018.

[2] F. Ferrucci, Pro-active Dynamic Vehicle Routing: Real-time Control and Request-forecasting Approaches to Improve Customer Service. Springer Science & Business Media, 2013.

[3] M. of Transport, "Transportation in canada 2015, overview report," Transport Canada, 2015.

[4] J. Taylor, K. Casavant, D. Moore, J. Sage et B. Ivanov, "The economic impact of increased congestion for freight dependent businesses in washington state," Washington State Department of Transportation, 2012.

[5] M. Chen et S. I. Chien, "Dynamic freeway travel-time prediction with probe vehicle data: Link based versus path based," Transportation Research Record, vol. 1768, n°. 1, p. 157–161, 2001.

[6] S. I.-J. Chien et C. M. Kuchipudi, "Dynamic travel time prediction with real-time and historic data," Journal of transportation engineering, vol. 129, n°. 6, p. 608–616, 2003.

[7] X. Zhang et J. A. Rice, "Short-term travel time prediction," Transportation Research Part C: Emerging Technologies, vol. 11, n°. 3-4, p. 187–210, 2003.

[8] C. I. van Hinsbergen et J. Van Lint, "Bayesian combination of travel time prediction models," Transportation Research Record, vol. 2064, n°. 1, p. 73–80, 2008.

[9] J. Kwon, B. Coifman et P. Bickel, "Day-to-day travel-time trends and travel-time prediction from loop-detector data," Transportation Research Record, vol. 1717, n°. 1, p. 120–129, 2000.

[10] H. Chen, S. Grant-Muller, L. Mussone et F. Montgomery, "A study of hybrid neural network approaches and the effects of missing data on traffic forecasting," Neural Computing & Applications, vol. 10, n°. 3, p. 277–286, 2001.

[11] M. Moniruzzaman, H. Maoh et W. Anderson, "Short-term prediction of border crossing time and traffic volume for commercial trucks: A case study for the Ambassador

bridge," Transportation Research Part C: Emerging Technologies, vol. 63, p. 182–194, 2016.

[12] X. Ma, Z. Tao, Y. Wang, H. Yu et Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," Transportation Research Part C: Emerging Technologies, vol. 54, p. 187–197, 2015.

[13] C. Malandraki et M. Daskin, "Time dependent vehicle routing problems: formulations, properties and heuristic algorithms," Transportation Science, vol. 26, nº. 3, p. 185–200, 1992.

[14] S. Ichoua, M. Gendreau et J.-Y. Potvin, "Vehicle dispatching with time-dependent travel times," European Journal of Operational Research, vol. 144, nº. 2, p. 379–396, 2003.

[15] A. Haghani et S. Jung, "A dynamic vehicle routing problem with time-dependent travel times," Computers & Operations Research, vol. 32, nº. 11, p. 2959–2986, 2005.

[16] S. Jung et A. Haghani, "Genetic algorithm for the time-dependent vehicle routing problem," Transportation Research Record: Journal of the Transportation Research Board, nº. 1771, p. 164–171, 2001.

[17] A. Donati, R. Montemanni, N. Casagrande, A. Rizzoli et L. Gambardella, "Time dependent vehicle routing problem with a multi ant colony system," European Journal of Operational Research, vol. 185, nº. 3, p. 1174–1191, 2008.

[18] M. Figliozzi, "The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics," Transportation Research Part E: Logistics and Transportation Review, vol. 48, nº. 3, p. 616–636, 2012.

[19] D. Taş, N. Dellaert, T. van Woensel et T. de Kok, "The time-dependent vehicle routing problem with soft time windows and stochastic travel times," Transportation Research Part C: Emerging Technologies, vol. 48, p. 66–83, 2014.

[20] S. Dabia, S. Ropke, T. Van Woensel et T. De Kok, "Branch and price for the time-dependent vehicle routing problem with time windows," Transportation Science, vol. 47, nº. 3, p. 380–396, 2013.

[21] M. Setak, M. Habibi, H. Karimi et M. Abedzadeh, "A time-dependent vehicle routing problem in multigraph with FIFO property," Journal of Manufacturing Systems, vol. 35, p. 37–45, 2015.

[22] D. S. Lai, O. C. Demirag et J. M. Leung, "A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph," Transportation Research Part E: Logistics and Transportation Review, vol. 86, p. 32–52, 2016.

[23] H. Wang et Y. Lee, "Two-stage particle swarm optimization algorithm for the time dependent alternative vehicle routing problem," Journal of Applied & Computational Mathematics, vol. 3, nº. 4, p. 1–9, 2014.

[24] S. Mancini, "Time dependent travel speed vehicle routing and scheduling on a real road network: the case of Torino," Transportation Research Procedia, vol. 3, p. 433–441, 2014.

[25] Y. Huang, L. Zhao, T. Van Woensel et J.-P. Gross, "Time-dependent vehicle routing problem with path flexibility," Transportation Research Part B: Methodological, vol. 95, p. 169–195, 2017.

[26] H. Ben Ticha, "Vehicle routing problems with Road-Network Information," Thèse de doctorat, Université Clermont Auvergne, 2017.

[27] B. Fleischmann, S. Gnutzmann et E. Sandvoß, "Dynamic vehicle routing based on online traffic information," Transportation science, vol. 38, nº. 4, p. 420–433, 2004.

[28] E. Taniguchi et H. Shimamoto, "Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times," Transportation Research Part C: Emerging Technologies, vol. 12, nº. 3-4, p. 235–250, 2004.

[29] A. Haghani et S. Jung, "A dynamic vehicle routing problem with time-dependent travel times," Computers & operations research, vol. 32, nº. 11, p. 2959–2986, 2005.

[30] J.-Y. Potvin, Y. Xu et I. Benyahia, "Vehicle routing and scheduling with dynamic travel times," Computers & Operations Research, vol. 33, nº. 4, p. 1129–1137, 2006.

[31] S. Lorini, J.-Y. Potvin et N. Zufferey, "Online vehicle routing and scheduling with dynamic travel times," Computers & Operations Research, vol. 38, nº. 7, p. 1086–1090, 2011.

[32] J. Respen, N. Zufferey et J.-Y. Potvin, Impact of online tracking on a vehicle routing problem with dynamic travel times. CIRRELT, 2014.

[33] T. Van Woensel, L. Kerbache, H. Peremans et N. Vandaele, "Vehicle routing with dynamic travel times: A queueing approach," European journal of operational research, vol. 186, nº. 3, p. 990–1007, 2008.

[34] C. Van Hinsbergen, J. Van Lint et F. Sanders, "Short term traffic prediction models," dans Proceedings of the 14th World Congress on Intelligent Transportation Systems (ITS), Bejing, China, 2007.

[35] B. Smith, B. Williams et R. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," Transportation Research Part C: Emerging Technologies, vol. 10, n°. 4, p. 303–321, 2002.

[36] C.-H. Wu, J.-M. Ho et D.-T. Lee, "Travel-time prediction with support vector regression," IEEE Transactions on Intelligent Transportation Systems, vol. 5, n°. 4, p. 276–281, 2004.

[37] R. Kalman et R. Bucy, "New results in linear filtering and prediction theory," Journal of Basic Engineering, vol. 83, n°. 1, p. 95–108, 1961.

[38] S. J. Julier et J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," dans Signal processing, sensor fusion, and target recognition VI, vol. 3068. International Society for Optics and Photonics, 1997, p. 182–193.

[39] Y. Wang et M. Papageorgiou, "Real-time freeway traffic state estimation based on extended Kalman filter: A general approach," Transportation Research Part B: Methodological, vol. 39, n°. 2, p. 141–167, 2005.

[40] J. Van Lint, "Online learning solutions for freeway travel time prediction," IEEE Transactions on Intelligent Transportation Systems, vol. 9, n°. 1, p. 38–47, 2008.

[41] C. Antoniou, M. Ben-Akiva et H. Koutsopoulos, "Nonlinear Kalman filtering algorithms for on-line calibration of dynamic traffic assignment models," IEEE Transactions on Intelligent Transportation Systems, vol. 8, n°. 4, p. 661–670, 2007.

[42] H. Jula, M. Dessouky et P. Ioannou, "Real-time estimation of travel times along the arcs and arrival times at the nodes of dynamic stochastic networks," IEEE Transactions on Intelligent Transportation Systems, vol. 9, n°. 1, p. 97–110, 2008.

[43] M. Hamed, H. Al-Masaeid et Z. Said, "Short-term prediction of traffic volume in urban arterials," Journal of Transportation Engineering, vol. 121, n°. 3, p. 249–254, 1995.

[44] G. E. Box, G. M. Jenkins, G. C. Reinsel et G. M. Ljung, Time series analysis: forecasting and control. John Wiley & Sons, 2015.

[45] B. Williams et L. Hoel, “Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results,” Journal of Transportation Engineering, vol. 129, nº. 6, p. 664–672, 2003.

[46] J. Guo, “Adaptive Estimation and Prediction of Univariate Vehicular Traffic Condition Series,” Ph.D. Thesis, North Carolina State University, 2005.

[47] V. Vapnik, “An overview of statistical learning theory,” IEEE Transactions on Neural Networks, vol. 10, nº. 5, p. 988–999, 1999.

[48] V. Vladimir, The Nature of Statistical Learning Theory. Springer Heidelberg, 1995.

[49] J. Wang et Q. Shi, “Short-term traffic speed forecasting hybrid model based on chaos–wavelet analysis-support vector machine theory,” Transportation Research Part C: Emerging Technologies, vol. 27, p. 219–232, 2013.

[50] Z. Yang, D. Mei, Q. Yang, H. Zhou et X. Li, “Traffic flow prediction model for large-scale road network based on cloud computing,” Mathematical Problems in Engineering, 2014.

[51] E. Vlahogianni, M. G. Karlaftis et J. Golias, “Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach,” Transportation Research Part C: Emerging Technologies, vol. 13, nº. 3, p. 211–234, 2005.

[52] A. Habtie, A. Abraham et D. Midekso, “Artificial neural network based real-time urban road traffic state estimation framework,” dans Computational Intelligence in Wireless Sensor Networks. Springer, 2017, p. 73–97.

[53] Y. Tian et L. Pan, “Predicting short-term traffic flow by long short-term memory recurrent neural network,” dans IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity). IEEE, 2015, p. 153–158.

[54] R. Fu, Z. Zhang et L. Li, “Using LSTM and GRU neural network methods for traffic flow prediction,” dans Youth Academic Annual Conference of the Chinese Association of Automation. IEEE, 2016, p. 324–328.

[55] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk et Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” arXiv preprint arXiv:1406.1078, 2014.

[56] G. Davis et N. Nihan, “Nonparametric regression and short-term freeway traffic forecasting,” Journal of Transportation Engineering, vol. 117, nº. 2, p. 178–188, 1991.

[57] B. Smith, “Forecasting Freeway Traffic Flow for Intelligent Transportation Systems Application,” Thèse de doctorat, Charlottesville, VA, USA, 1995.

[58] B. Smith et M. Demetsky, “Traffic flow forecasting: comparison of modeling approaches,” Journal of Transportation Engineering, vol. 123, nº. 4, p. 261–266, 1997.

[59] A. Ermagun et D. Levinson, “Spatiotemporal traffic forecasting: Review and proposed directions,” Transport Reviews, p. 1–29, 2018.

[60] M. Hashemi et H. Karimi, “A weight-based map-matching algorithm for vehicle navigation in complex urban networks,” Journal of Intelligent Transportation Systems, vol. 20, nº. 6, p. 573–590, 2016.

[61] J. L. Schafer, “Multiple imputation: a primer,” Statistical methods in medical research, vol. 8, nº. 1, p. 3–15, 1999.

[62] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” dans Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1. University of California Press, 1967, p. 281–297.

[63] B. Frey et D. Dueck, “Clustering by passing messages between data points,” Science, vol. 315, nº. 5814, p. 972–976, 2007.

[64] K. Fukunaga et L. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” IEEE Transactions on Information Theory, vol. 21, nº. 1, p. 32–40, 1975.

[65] P. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” Journal of Computational and Applied Mathematics, vol. 20, p. 53–65, 1987.

[66] T. Calinski et J. Harabasz, “A dendrite method for cluster analysis,” Communications in Statistics, vol. 3, nº. 1, p. 1–27, 1974.

[67] D. Rubin, Multiple Imputation for Nonresponse in Surveys. John Wiley & Sons, 2004, vol. 81.

[68] R. Little et D. Rubin, Statistical Analysis with Missing Data. John Wiley & Sons, 2019, vol. 793.

[69] S. Buuren et K. Groothuis-Oudshoorn, “MICE: Multivariate imputation by chained equations in R,” Journal of Statistical Software, vol. 45, nº. 3, 2011.

[70] D. Stekhoven et P. Bühlmann, “Missforest-Non-parametric missing value imputation for mixed-type data,” Bioinformatics, vol. 28, nº. 1, p. 112–118, 2011.

[71] J. Honaker, G. King et M. Blackwell, “Amelia II: A program for missing data,” Journal of Statistical Software, vol. 45, nº. 7, p. 1–47, 2011.

[72] L. Breiman, “Random forests,” Machine Learning, vol. 45, nº. 1, p. 5–32, 2001.

[73] A. Liaw et M. Wiener, “Classification and regression by randomforest,” R news, vol. 2, nº. 3, p. 18–22, 2002.

[74] B. Efron et R. Tibshirani, An introduction to the bootstrap. CRC press, 1994.

[75] A. Dempster, N. Laird et D. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” Journal of the Royal Statistical Society. Series B, p. 1–38, 1977.

[76] S. Hochreiter et J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, nº. 8, p. 1735–1780, 1997.

[77] R. Pascanu, T. Mikolov et Y. Bengio, “On the difficulty of training recurrent neural networks,” dans International Conference on Machine Learning, 2013, p. 1310–1318.

[78] K. Greff, R. Srivastava, J. Koutník, B. Steunebrink et J. Schmidhhuber, “LSTM: A search space odyssey,” IEEE Trans. on Neural Networks and Learning Systems, vol. 28, nº. 10, p. 2222–2232, 2017.

[79] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot et E. Duchesnay, “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, p. 2825–2830, 2011.

[80] X. Glorot et Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” dans Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, p. 249–256.

[81] J. Bergstra, R. Bardenet, Y. Bengio et B. Kégl, “Algorithms for hyper-parameter optimization,” dans Advances in Neural Information Processing Systems 24, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira et K. Weinberger, édit. Curran Associates, Inc., 2011, p. 2546–2554.

[82] J. Snoek, H. Larochelle et R. Adams, “Practical Bayesian optimization of machine learning algorithms,” dans Advances in Neural Information Processing Systems, 2012, p. 2951–2959.

[83] J. Bergstra et Y. Bengio, “Random search for hyper-parameter optimization,” Journal of Machine Learning Research, vol. 13, p. 281–305, 2012.

[84] D. Kingma et J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[85] S. Ruder, “An overview of gradient descent optimization algorithms,” arXiv preprint arXiv:1609.04747, 2016.

[86] J. Schafer et M. Olsen, “Multiple imputation for multivariate missing-data problems: A data analyst’s perspective,” Multivariate Behavioral Research, vol. 33, n°. 4, p. 545–571, 1998.

[87] J. Graham, A. Olchowski et T. Gilreath, “How many imputations are really needed? Some practical clarifications of multiple imputation theory,” Prevention Science, vol. 8, n°. 3, p. 206–213, 2007.

[88] J. Schafer et J. Graham, “Missing data: our view of the state of the art.” Psychological Methods, vol. 7, n°. 2, p. 147, 2002.

[89] I. Sutskever, “Training Recurrent Neural Networks,” Ph D. Thesis, University of Toronto, Toronto, Canada, 2013.

[90] R. Baldacci, A. Mingozzi et R. Roberti, “Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints,” European Journal of Operational Research, vol. 218, n°. 1, p. 1–6, 2012.

[91] O. Bräysy et M. Gendreau, “Vehicle routing problem with time windows, Part I: Route construction and local search algorithms,” Transportation Science, vol. 39, n°. 1, p. 104–118, 2005.

[92] O. Bräysy et M. Gendreau, “Vehicle routing problem with time windows, Part II: Metaheuristics,” Transportation Science, vol. 39, n°. 1, p. 119–139, 2005.

[93] M. Gendreau, G. Ghiani et E. Guerriero, “Time-dependent routing problems: A review,” Computers & Operations Research, vol. 64, p. 189–197, 2015.

[94] B. Fleischmann, M. Gietz et S. Gnutzmann, “Time-varying travel times in vehicle routing,” Transportation Science, vol. 38, n°. 2, p. 160–173, 2004.

[95] H. Ben Ticha, N. Absi, D. Feillet, A. Quilliot et T. van Woensel, “Time-dependent vehicle routing problem with time windows and road network information,” Rapport technique.

[96] T. Garaix, C. Artigues, D. Feillet et D. Josselin, “Vehicle routing problems with alternative paths: An application to on-demand transportation,” European Journal of Operational Research, vol. 204, n°. 1, p. 62–75, 2010.

[97] F. Glover, “Tabu search–Part I,” ORSA Journal on Computing, vol. 1, n°. 3, p. 190–206, 1989.

[98] G. Laporte, S. Ropke et T. Vidal, “Chapter 4: Heuristics for the vehicle routing problem,” dans Vehicle Routing: Problems, Methods, and Applications, Second Edition. SIAM, 2014, p. 87–116.

[99] É. Taillard, P. Badeau, M. Gendreau, F. Guertin et J.-Y. Potvin, “A tabu search heuristic for the vehicle routing problem with soft time windows,” Transportation Science, vol. 31, n°. 2, p. 170–186, 1997.

[100] A. Letchford, S. Nasiri et D. Theis, “Compact formulations of the Steiner traveling salesman problem and related problems,” European Journal of Operational Research, vol. 228, n°. 1, p. 83–92, 2013.

[101] G. Dantzig et J. Ramser, “The truck dispatching problem,” Management Science, vol. 6, n°. 1, p. 80–91, 1959.

[102] P. Toth et D. Vigo, Vehicle Routing: Problems, Methods, and Applications. SIAM, 2014.

[103] H. Ben Ticha, N. Absi, D. Feillet, A. Quilliot et T. Van Woensel, “A branch-and-price algorithm for the vehicle routing problem with time windows on a road network,” Networks, vol. 73, n°. 4, p. 401–417, 2019.

[104] M. Gmira, M. Gendreau, A. Lodi et J.-Y. Potvin, “Tabu search for the time-dependent vehicle routing problem with time windows on a road network,” CIRRELT-2019-32, Montréal, Canada, Rapport technique, 2019.

[105] H. Psaraftis, “Dynamic vehicle routing problems,” dans Vehicle routing: Methods and Studies, B. Golden et A. Assad, édit. North-Holland, 1988.

[106] H. Psaraftis, M. Wen et C. Kontovas, “Dynamic vehicle routing problems: Three decades and counting,” Networks, vol. 67, n°. 1, p. 3–31, 2016.

[107] T. Bektaş, P. Repoussis et C. Tarantilis, “Chapter 11: Dynamic Vehicle Routing Problems,” dans Vehicle Routing: Problems, Methods, and Applications, Second Edition. SIAM, 2014, p. 299–347.

[108] V. Pillac, M. Gendreau, C. Guéret et A. Medaglia, “A review of dynamic vehicle routing problems,” European Journal of Operational Research, vol. 225, nº. 1, p. 1–11, 2013.

[109] S. Ichoua, M. Gendreau et J.-Y. Potvin, “Planned route optimization for real-time vehicle routing,” dans Fleet Management and Logistics, T. Crainic et G. Laporte, édit. Springer, 2007, p. 1–18.

[110] M. Khouadjia, B. Sarasola, E. Alba, E.-G. Talbi et L. Jourdan, “Metaheuristics for dynamic vehicle routing,” dans Metaheuristics for Dynamic Optimization. Springer, 2013, p. 265–289.

[111] M. Gendreau, F. Guertin, J.-Y. Potvin et É. Taillard, “Parallel tabu search for real-time vehicle routing and dispatching,” Transportation Science, vol. 33, nº. 4, p. 381–390, 1999.

[112] S. Ichoua, M. Gendreau et J.-Y. Potvin, “Diversion issues in real-time vehicle dispatching,” Transportation Science, vol. 34, nº. 4, p. 426–438, 2000.

[113] R. Montemanni, L. Gambardella, A. Rizzoli et A. Donati, “Ant colony system for a dynamic vehicle routing problem,” Journal of Combinatorial Optimization, vol. 10, nº. 4, p. 327–343, 2005.

[114] V. Pureza et G. Laporte, “Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows,” INFOR: Information Systems and Operational Research, vol. 46, nº. 3, p. 165–175, 2008.

[115] S. Mitrović-Minić, R. Krishnamurti et G. Laporte, “Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows,” Transportation Research Part B, vol. 38, nº. 8, p. 669–685, 2004.

[116] R. Bent et P. Van Hentenryck, “Scenario-based planning for partially dynamic vehicle routing with stochastic customers,” Operations Research, vol. 52, nº. 6, p. 977–987, 2004.

[117] S. Binart, P. Dejax, M. Gendreau et F. Semet, “A 2-stage method for a field service routing problem with stochastic travel and service times,” Computers & Operations Research, vol. 65, p. 64–75, 2016.

[118] I. Or, “Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking,” PhD thesis, Department of Industrial Engineering and Management Science, Northwestern University, U.S.A., 1976.

[119] M. Gmira, M. Gendreau, A. Lodi et J.-Y. Potvin, “Travel speed prediction based on learning methods for home delivery,” CIRRELT-2018-46, Montréal, Canada, Rapport technique, 2018.

[120] A. Letchford, S. Nasiri et A. Oukil, “Pricing routines for vehicle routing with time windows on road networks,” Computers & Operations Research, vol. 51, p. 331–337, 2014.

[121] L. He, M. Trépanier et B. Agard, “Comparing time series segmentation methods for the analysis of transportation patterns with smart card data,” CIRRELT, Centre interuniversitaire de recherche sur les réseaux d’entreprise . . . , Rapport technique, 2017.